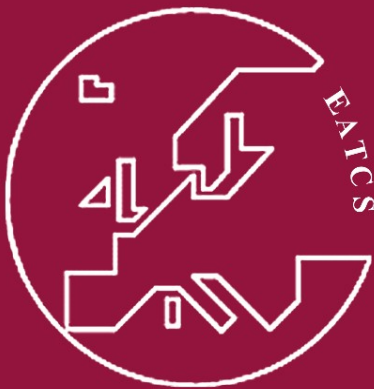


Lars Arge  
Christian Cachin  
Tomasz Jurdziński  
Andrzej Tarlecki (Eds.)

LNCS 4596

# Automata, Languages and Programming

34th International Colloquium, ICALP 2007  
Wrocław, Poland, July 2007  
Proceedings



 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Moshe Y. Vardi

*Rice University, Houston, TX, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

Lars Arge Christian Cachin  
Tomasz Jurdziński Andrzej Tarlecki (Eds.)

# Automata, Languages and Programming

34th International Colloquium, ICALP 2007  
Wrocław, Poland, July 9-13, 2007  
Proceedings

 Springer

## Volume Editors

Lars Arge

University of Aarhus, Department of Computer Science  
IT-Parken, Aabogade 34, 8200 Aarhus N, Denmark  
E-mail: large@daimi.au.dk

Christian Cachin

IBM Research, Zurich Research Laboratory  
Säumerstrasse 4, 8803 Rüschlikon, Switzerland  
E-mail: cca@zurich.ibm.com

Tomasz Jurdziński

University of Wrocław, Institute of Computer Science  
ul. Joliot-Curie 15, 50-383 Wrocław, Poland  
E-mail: tju@ii.uni.wroc.pl

Andrzej Tarlecki

University of Warsaw, Institute of Informatics  
ul. Banacha 2, 02-097 Warsaw, Poland  
E-mail: tarlecki@mimuw.edu.pl

Library of Congress Control Number: 2007929786

CR Subject Classification (1998): F, D, C.2-3, G.1-2, I.3, E.1-2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743

ISBN-10 3-540-73419-8 Springer Berlin Heidelberg New York

ISBN-13 978-3-540-73419-2 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2007

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper SPIN: 12085491 06/3180 5 4 3 2 1 0

# Preface

The 34th International Colloquium on Automata, Languages and Programming (ICALP 2007) was held in Wrocław, Poland, on July 9–13, 2007. This volume contains all papers selected for presentation at ICALP 2007. The conference program also included excellent invited lectures by Bernard Chazelle (Princeton), Ivan Damgård (Aarhus), Fedor Fomin (Bergen), Gordon Plotkin (Edinburgh), Michael O. Rabin (Harvard), and Fred Schneider (Cornell); some of these were accompanied by papers that are included in this volume as well. In addition, a paper by Michael O. Rabin, who gave a joint LICS/ICALP invited lecture, is included in the proceedings of LICS 2007.

ICALP constitutes a series of annual conferences of the European Association for Theoretical Computer Science (EATCS), the first one of which took place in 1972. This year, the ICALP program consisted of three tracks, following the model established with ICALP 2005: Track A focusing on algorithms, automata, complexity and games; Track B focusing on logic, semantics and theory of programming; and Track C focusing on security and cryptography foundations.

In response to the call for papers, the Program Committee received 242 submissions, of which 149 for Track A, 59 for Track B and 34 for Track C. The Program Committee selected 76 papers for inclusion in the scientific program, in particular, 41 papers for Track A, 24 papers for Track B, and 11 papers for Track C. The selection was made by the Program Committee based on originality, quality, and relevance to theoretical computer science.

Even though the total number of submissions was somewhat lower than in the previous years, the overall quality of the submissions was very good indeed. During the highly competitive selection process — as tough as it is expected to be at ICALP — many good submissions had to be rejected and the outcome resulted in a very interesting overall program.

ICALP 2007 was held in conjunction with the 22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007), the 9th ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP 2007), and Logic Colloquium 2007. The following workshops were held as satellite events of ICALP 2007 and LICS 2007:

- DCM 2007: 3rd International Workshop on Development of Computational Models;
- FCS-ARSPA 2007: Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis;
- GOCP 2007: International Workshop on Group-Oriented Cryptographic Protocols;
- LCC 2007: 9th International Workshop on Logic and Computational Complexity;
- PAuL 2007: International Workshop on Probabilistic Automata and Logics;

- Reasoning about and with Games 2007;
- SOS 2007: 4th Workshop on Structural Operational Semantics;
- TRSH 2007: Theory of Randomized Search Heuristic;
- TMCNAA 2007: Workshop on Traced Monoidal Categories, Network Algebras, and Applications; and
- WCAN 2007: 3rd Workshop on Cryptography for Ad-hoc Networks.

Moreover, the 3rd International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2007) was held in Wrocław at the same time.

We wish to thank all authors who submitted extended abstracts for consideration, the Program Committee for their hard work, and the referees who assisted the Program Committee in the evaluation process.

We are grateful to the University of Wrocław for hosting ICALP 2007 and to the Organizing Committee at the Institute of Computer Science, chaired by Jerzy Marcinkowski, for mastering the logistics of ICALP 2007 and the concurrent events. We also thank the University of Wrocław and the City of Wrocław for their financial support.

We gratefully acknowledge the use of the following two conference management systems: EasyChair (Tracks A and B) by Andrei Voronkov and the Web Submission and Review Software by Shai Halevi (Track C).

April 2007

Lars Arge  
Christian Cachin  
Tomasz Jurdziński  
Andrzej Tarlecki

# Conference Organization

## Program Committee

### Track A

Susanne Albers, Universität Freiburg, Germany  
Lars Arge, University of Aarhus, Denmark (Chair)  
James Aspnes, Yale University, USA  
Yossi Azar, Microsoft Research, USA and Tel-Aviv University, Israel  
Joan Boyar, University of Southern Denmark, Denmark  
Richard Cole, New York University, USA  
Camil Demetrescu, University of Rome La Sapienza, Italy  
Xiaotie Deng, City University of Hong Kong, Hong Kong  
Thomas Erlebach, University of Leicester, UK  
Ricard Gavaldà, Universitat Politècnica de Catalunya, Spain  
Loukas Georgiadis, HP, USA  
Sariel Har-Peled, University of Illinois at Urbana-Champaign, USA  
Markus Holzer, Technische Universität München, Germany  
Piotr Indyk, Massachusetts Institute of Technology, USA  
Ming-Yang Kao, Northwestern University, USA  
Marc van Kreveld, Utrecht University, The Netherlands  
Ulrich Meyer, MPI-INF Saarbrücken, Germany  
Michael Mitzenmacher, Harvard University, USA  
Branislav Rován, Comenius University, Slovakia  
Jan Arne Telle, University of Bergen, Norway  
Jacobo Torán, Universität Ulm, Germany  
Norbert Zeh, Dalhousie University, Canada

### Track B

Roland Backhouse, University of Nottingham, UK  
Pierpaolo Degano, Università di Pisa, Italy  
Bengt Jonsson, Uppsala University, Sweden  
Christoph Koch, Universität des Saarlandes, Saarbrücken, Germany  
Marta Kwiatkowska, University of Birmingham, UK  
Martin Lange, University of Aarhus, Denmark  
Pierre Lescanne, ENS Lyon, France  
Madhusudan Parthasarathy, University of Illinois, Urbana-Champaign, USA  
Eugenio Moggi, Università di Genova, Italy  
Lawrence Moss, Indiana University, Bloomington, USA  
Andrzej Murawski, Oxford University, UK

## VIII Organization

Fernando Orejas, UPC, Barcelona, Spain  
Doron Peled, University of Warwick, UK  
Davide Sangiorgi, Università di Bologna, Italy  
Thomas Schwentick, Universität Dortmund, Germany  
Wolfgang Thomas, RWTH Aachen, Germany  
Lidia Tendera, Opole University, Poland  
Andrzej Tarlecki, Warsaw University, Poland (Chair)  
Frits Vaandrager, Radboud University Nijmegen, The Netherlands  
Rob van Glabbeek, NICTA, Sydney, Australia

### Track C

Michael Backes, Saarland University, Germany  
David Basin, ETH Zürich, Switzerland  
Michele Boreale, Università di Firenze, Italy  
Christian Cachin, IBM Research, Switzerland (Chair)  
Ran Canetti, IBM Research, USA  
Véronique Cortier, LORIA, CNRS, France  
Stefan Dziembowski, Warsaw University, Poland  
and University of Rome La Sapienza, Italy  
Cédric Fournet, Microsoft Research, UK  
Jonathan Katz, University of Maryland, USA  
Eike Kiltz, CWI, The Netherlands  
Eyal Kushilevitz, Technion, Israel  
Anna Lysyanskaya, Brown University, USA  
Jesper Nielsen, University of Aarhus, Denmark  
Rafael Pass, Cornell University, USA  
Giuseppe Persiano, University of Salerno, Italy  
Krzysztof Pietrzak, ENS, France  
Leonid Reyzin, Boston University, USA  
Mark Ryan, University of Birmingham, UK  
David Sands, Chalmers University of Technology, Sweden  
Abhi Shelat, IBM Research, Switzerland

### Local Organizing Committee

Jerzy Marcinkowski (Conference Chair)  
Marcin Bieńkowski (Workshop Chair)  
Artur Jeż  
Tomasz Jurdziński  
Emanuel Kieroński  
Krzysztof Loryś  
Aleksander Mądry



## Sponsoring Institutions

The University of Wrocław  
The City of Wrocław

## External Reviewers

Pavan Aduri, Deepak Ajwani, Natasha Alechina, Jesus Almendros, Ernst Althaus, Andris Ambainis, Elliot Anshelevich, Pablo Arrighi, Vikraman Arvind, Roland Axelsson, David Bader, Patrick Baillot, Elisa Baniassad, Nikhil Bansal, Massimo Bartoletti, Surender Baswana, Mark de Berg, Henrik Bjorklund, Manuel Bodirsky, Hans Bodlaender, Beate Bollig, Filippo Bonchi, Maria Luisa Bonet, Henning Bordihn, Magdalene G. Borgelt, Xavier Boyen, Claus Brabrand, Felix Brandt, Gerth Brodal, Joachim Buhmann, Tian-Ming Bu, Nadia Busi, Jin-Yi Cai, Arnaud Carayol, Josep Carmona, Anton Cerny, Timothy Chan, Kevin Chang, Witold Charatonik, Shuchi Chawla, Chandra Chekuri, Hubie Chen, Jianer Chen, Jing Chen, Xi Chen, Jacek Chrzęszcz, Jacek Cichoń, David Cock, Tom Coleman, Giovanni Conforti, Ricardo Corin, Andrea Corradini, Anne Cregan, Artur Czumaj, Ivan Damgård, Pous Damien, Nenad Dedić, Stefan Dobrev, Sebastian Dörn, Petros Drineas, Pavol Duris, Jana Dvorakova, Faith Ellen, Amir Epstein, Leah Epstein, Kousha Etessami, Rolf Fagerberg, Regan Faretton, Lene Favroldt, Uriel Feige, Michael Fellows, Jose Fernandez, Henning Fernau, Gian Luigi Ferrari, Felix Fischer, Marc Fischlin, Lisa Fleischer, Fedor Fomin, Lance Fortnow, Dimitris Fotakis, Paolo Franciosa, Stefan Funke, J. Gabarro, Clemente Galdi, Iftah Gamzu, Naveen Garg, Dmitry Gavinsky, Cyril Gavoille, Assefaw Gebremedhin, Blaise Genest, Leszek Gaşieniec, Paul Goldberg, Alexander Golynski, Fred Green, Martin Green, Hermann Gruber, Dimitar Guelev, Jiong Guo, Mohammadtaghi Hajiaghayi, Jason Hartline, Herman Haverkort, Karmit Hazai, Sun He, Rogardt Heldal, Pavol Hell, Danny Hendler, Peter Hertling, Yoram Hirshfeld, Tomas Holan, Peter Høyer, Chun-Yuan Hsiao, Li-Sha Huang, Nicole Immerlica, Kazuo Iwama, Shahid Jabbar, Kamal Jain, Jesper Jansson, Peter Jeavons, Jan Johannsen, Marcin Jurdziński, Matthieu Kaczmarek, Lukasz Kaiser, Bhavana Kanukurthi, Ming-Yang Kao, George Karakostas, Juhani Karhumaki, Wong Karianto, Branislav Katreniak, Irit Katriel, Emanuel Kieroński, Hartmut Klauck, Philip Klein, Bartek Klin, Jochen Könemann, Chiu-Yuen Koo, Sven Kosub, Michal Koucký, Annamaria Kovacs, Rastislav Královič, Richard Královič, Dieter Kratsch, Hugo Krawczyk, Steve Kremer, Michal Kunc, Orna Kupferman, Martin Kutrib, Pascal Lafourcade, Mark Lanthier, Sophie Laplante, Sławomir Lasota, Stefano Leonardi, Jerome Leroux, Pierre Lescanne, Martin Leucker, Minming Li, Yehuda Lindell, Haowen Liu, Christof Loeding, Maarten Löffler, Francisco Lopez-Fraguas, Alex Lopez-Ortiz, Michele Loreti, Hsueh-I Lu, Jun Luo, Yuh-Dauh Lyuu, Edita Macajova, Damiano Macedonio, Ian Mackie, Meena Mahajan, Mohammad Mahdian, Anil Maheshwari, Yishay Mansour, Jerzy Marcinkowski, Wim Martens, Narciso Marti-Oliet, Dániel Marx, Claire Mathieu, Neel Mathur, Marios Mavronikolas,

Andrew McGregor, Annabelle McIver, Pierre McKenzie, Klaus Meer, Daniel Meister, Dieter van Melkebeek, Maria-Chiara Meo, Xavier Messeguer, Jochen Messner, Julian Mestre, Paolo Milazzo, Vahab Mirrokni, Michael Mislove, Takunari Miyazaki, Arjan Mooij, Shlomo Moran, Carroll Morgan, Larry Moss, Herve Moulin, Fabian Mueller, Aybek Mukhamedov, Ian Munro, Dejan Nickovic, Rolf Niedermeier, Gethin Norman, Marc Noy, Jan Obdrzalek, Kazuhiro Ogata, Alexander Okhotin, Shien-Jin Ong, Vitaly Osipov, Martin Pal, Adriana Palacio, Matthew Parker, Joachim Parrow, Wiesław Pawłowski, Doron Peled, Gerardo Pelosi, Mathew Penrose, Ion Petre, Jean-Eric Pin, Tomas Plachetka, Chung Keung Poon, Giuseppe Prencipe, Andrzej Proskurowski, Qi Qi, Yuri Rabinovich, Balaji Raghavachari, Daniel Raible, Rajeev Raman, Oded Regev, Klaus Reinhardt, Günter Rote, Terry G. Rudolph, Scott Russell, Daniil Ryabko, Ondrej Rypacek, Kunihiko Sadakane, Konstantinos Sagonas, Peter Sanders, Miklos Santa, Srinivasa Rao Satti, Ulrike Sattler, Saket Saurabh, Andrea Schalk, Christian Schallhart, Siu-Wing Scheng, Georg Schnitger, Uwe Schoening, Tom Schrijvers, Detlef Seese, Maria J. Serna, Asaf Shapira, Vitaly Shmatikov, Rodrigo Silveira, Riccardo Silvestri, Michiel Smid, Roberto Solis-Oba, Bettina Speckmann, Paola Spoletini, Christoph Sprenger, Jeremy Sproston, Jiri Srba, Ludwig Staiger, Martin Stanek, Daniel Stefankovic, Angelika Steger, Bernhard von Stengel, Marielle Stoelinga, Arne Storjohann, Wei Sun, Xiaoming Sun, Xiaosun Sun, Wing Kin Sung, Frédéric Sur, Chaitanya Swamy, Claude Tardif, Michael Tautschnig, Kavitha Telikepalli, Pascal Tesson, Thomas Thierauf, Wolfgang Thomas, Stravros Tripakis, Franco Turini, Dominique Unruh, Frits Vaandrager, Leslie Valiant, Daniele Varacca, Moshe Vardi, Stephen Vasavis, Muthuramakrishnan Venkitasubramaniam, Elad Verbin, Kumar Neeraj Verma, Björn Victor, German Vidal, Danny Vilenchik, Ivan Visconti, Mahesh Viswanathan, Heribert Vollmer, Imrich Vrt'o, Feng Wang, Lusheng Wang, Bogdan Warinschi, Osamu Watanabe, Brent Waters, John Watrous, Volker Weber, Benjamin Werner, Gordon Wilfong, Tony Wirth, Pierre Wolper, David Wood, David Woodruff, James Worrell, Fatos Xhafa, Guomin Yang, Wang Yi, Yitong Yin, Stanislav Zak, Michael Zakharyashev, Wenan Zang, Gianluigi Zavattaro, Konrad Zdanowski, Guochuan Zhang, Jun Zhang, Yunlei Zhao, Uri Zwick

# Table of Contents

## Invited Lectures

|  |    |
|--|----|
| Ushering in a New Era of Algorithm Design . . . . .                      | 1  |
| <i>Bernard Chazelle</i>  |    |
| A “Proof-Reading” of Some Issues in Cryptography . . . . .               | 2  |
| <i>Ivan Damgård</i>  |    |
| Credentials-Based Authorization: Evaluation and Implementation . . . . . | 12 |
| <i>Fred B. Schneider</i>   |    |
| Subexponential Parameterized Algorithms . . . . .                        | 15 |
| <i>Frederic Dorn, Fedor V. Fomin, and Dimitrios M. Thilikos</i>          |    |

## Session A1

|  |    |
|--|----|
| Competitive Algorithms for Due Date Scheduling . . . . .                   | 28 |
| <i>Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs</i>                        |    |
| Mechanism Design for Fractional Scheduling on Unrelated Machines . . . . . | 40 |
| <i>George Christodoulou, Elias Koutsoupias, and Annamária Kovács</i>       |    |

## Session A2

|  |    |
|--|----|
| Estimating Sum by Weighted Sampling . . . . .  | 53 |
| <i>Rajeev Motwani, Rina Panigrahy, and Ying Xu</i>                                     |    |
| Sampling Methods for Shortest Vectors, Closest Vectors and Successive Minima . . . . . | 65 |
| <i>Johannes Blömer and Stefanie Naewe</i>  |    |

## Session A3

|   |     |
|---|-----|
| Low Distortion Spanners . . . . .   | 78  |
| <i>Seth Pettie</i>  |     |
| Minimum Weight 2-Edge-Connected Spanning Subgraphs in Planar Graphs . . . . . | 90  |
| <i>André Berger and Michelangelo Grigni</i>                                   |     |
| Labeling Schemes for Vertex Connectivity . . . . .                            | 102 |
| <i>Amos Korman</i>  |     |

**Session A4**

Unbounded-Error One-Way Classical and Quantum Communication Complexity ..... 110  
*Kazuo Iwama, Harumichi Nishimura, Rudy Raymond, and Shigeru Yamashita*

A Lower Bound on Entanglement-Assisted Quantum Communication Complexity ..... 122  
*Ashley Montanaro and Andreas Winter*

Separating Deterministic from Nondeterministic NOF Multiparty Communication Complexity ..... 134  
*Paul Beame, Matei David, Toniann Pitassi, and Philipp Woelfel*

**Session A5**

An Optimal Decomposition Algorithm for Tree Edit Distance..... 146  
*Erik D. Demaine, Shay Mozes, Benjamin Rossman, and Oren Weimann*

On Commutativity Based Edge Lean Search..... 158  
*Dragan Bošnački, Edith Elkind, Blaise Genest, and Doron Peled*

Commitment Under Uncertainty: Two-Stage Stochastic Matching Problems ..... 171  
*Irit Katriel, Claire Kenyon-Mathieu, and Eli Upfal*

**Session A6**

On the Complexity of Hard-Core Set Constructions ..... 183  
*Chi-Jen Lu, Shi-Chun Tsai, and Hsin-Lung Wu*

Approximation by DNF: Examples and Counterexamples ..... 195  
*Ryan O’Donnell and Karl Wimmer*

Exotic Quantifiers, Complexity Classes, and Complete Problems ..... 207  
*Peter Bürgisser and Felipe Cucker*

**Session A7**

Online Conflict-Free Colorings for Hypergraphs ..... 219  
*Amotz Bar-Noy, Panagiotis Cheilaris, Svetlana Olonetsky, and Shakhar Smorodinsky*

Distributed Computing with Advice: Information Sensitivity of Graph Coloring ..... 231  
*Pierre Fraigniaud, Cyril Gavoille, David Ilcinkas, and Andrzej Pelc*

**Session C1**

|   |     |
|---|-----|
| Private Multiparty Sampling and Approximation of Vector<br>Combinations ..... | 243 |
| <i>Yuval Ishai, Tal Malkin, Martin J. Strauss, and Rebecca N. Wright</i>      |     |
| Constant-Round Private Database Queries .....                                 | 255 |
| <i>Nenad Dedic and Payman Mohassel</i>  |     |

**Session A8**

|  |     |
|--|-----|
| Universal Algebra and Hardness Results for Constraint Satisfaction<br>Problems ..... | 267 |
| <i>Benoît Larose and Pascal Tesson</i>   |     |
| On the Power of $k$ -Consistency .....   | 279 |
| <i>Albert Atserias, Andrei Bulatov, and Victor Dalmau</i>                            |     |
| Complexity of Propositional Proofs Under a Promise .....                             | 291 |
| <i>Nachum Dershowitz and Iddo Tzameret</i>   |     |

**Session C2**

|  |     |
|--|-----|
| Deterministic History-Independent Strategies for Storing Information<br>on Write-Once Memories ..... | 303 |
| <i>Tal Moran, Moni Naor, and Gil Segev</i>   |     |
| Trading Static for Adaptive Security in Universally Composable<br>Zero-Knowledge .....               | 316 |
| <i>Aggelos Kiayias and Hong-Sheng Zhou</i>   |     |
| A Characterization of Non-interactive Instance-Dependent<br>Commitment-Schemes (NIC) .....           | 328 |
| <i>Bruce Kapron, Lior Malka, and Venkatesh Srinivasan</i>  |     |

**Session A9**

|   |     |
|---|-----|
| Sharp Tractability Borderlines for Finding Connected Motifs in<br>Vertex-Colored Graphs ..... | 340 |
| <i>Michael R. Fellows, Guillaume Fertin, Danny Hermelin, and<br/>Stéphane Vialette</i>        |     |
| Parameterized Algorithms for Directed Maximum Leaf Problems .....                             | 352 |
| <i>Noga Alon, Fedor V. Fomin, Gregory Gutin,<br/>Michael Krivelevich, and Saket Saurabh</i>   |     |
| Parameterized Approximability of the Disjoint Cycle Problem .....                             | 363 |
| <i>Martin Grohe and Magdalena Grüber</i>  |     |
| Linear Problem Kernels for NP-Hard Problems on Planar Graphs .....                            | 375 |
| <i>Jiong Guo and Rolf Niedermeier</i>   |     |

**Session C3**

Private Locally Decodable Codes . . . . . 387  
*Rafail Ostrovsky, Omkant Pandey, and Amit Sahai*

Hash Functions in the Dedicated-Key Setting: Design Choices and  
MPP Transforms . . . . . 399  
*Mihir Bellare and Thomas Ristenpart*

Unrestricted Aggregate Signatures . . . . . 411  
*Mihir Bellare, Chanathip Namprempre, and Gregory Neven*

Ring Signatures of Sub-linear Size Without Random Oracles . . . . . 423  
*Nishanth Chandran, Jens Groth, and Amit Sahai*

**Session A10**

Balanced Families of Perfect Hash Functions and Their Applications . . . 435  
*Noga Alon and Shai Gutner*

An Exponential Improvement on the MST Heuristic for Minimum  
Energy Broadcasting in Ad Hoc Wireless Networks . . . . . 447  
*Ioannis Caragiannis, Michele Flammini, and Luca Moscardelli*

**Session B1**

Modular Algorithms for Heterogeneous Modal Logics . . . . . 459  
*Lutz Schröder and Dirk Pattinson*

Co-Logic Programming: Extending Logic Programming with  
Coinduction . . . . . 472  
*Luke Simon, Ajay Bansal, Ajay Mallya, and Gopal Gupta*

**Session C4**

Offline/Online Mixing . . . . . 484  
*Ben Adida and Douglas Wikström*

Fully Collusion Resistant Black-Box Traitor Revocable Broadcast  
Encryption with Short Private Keys . . . . . 496  
*Jun Furukawa and Nuttapon Attrapadung*

**Session A11**

Succinct Ordinal Trees Based on Tree Covering . . . . . 509  
*Meng He, J. Ian Munro, and S. Srinivasa Rao*

A Framework for Dynamizing Succinct Data Structures . . . . . 521  
*Ankur Gupta, Wing-Kai Hon, Rahul Shah, and Jeffrey Scott Vitter*

In-Place Suffix Sorting . . . . . 533  
*Gianni Franceschini and S. Muthukrishnan*

**Session B2**

|   |     |
|---|-----|
| Maximal Infinite-Valued Constraint Languages . . . . .              | 546 |
| <i>Manuel Bodirsky, Hubie Chen, Jan Kára, and Timo von Oertzen</i>  |     |
| Affine Systems of Equations and Counting Infinitary Logic . . . . . | 558 |
| <i>Albert Atserias, Andrei Bulatov, and Anuj Dawar</i>              |     |
| Boundedness of Monadic FO over Acyclic Structures . . . . .         | 571 |
| <i>Stephan Kreutzer, Martin Otto, and Nicole Schweikardt</i>        |     |

**Session A12**

|  |     |
|--|-----|
| Strong Price of Anarchy for Machine Load Balancing . . . . .   | 583 |
| <i>Amos Fiat, Haim Kaplan, Meital Levy, and Svetlana Olonetsky</i>                                     |     |
| Efficient Algorithms for Constant Well Supported Approximate<br>Equilibria in Bimatrix Games . . . . . | 595 |
| <i>Spyros C. Kontogiannis and Paul G. Spirakis</i>   |     |

**Session B3**

|  |     |
|--|-----|
| Equational Systems and Free Constructions . . . . .  | 607 |
| <i>Marcelo Fiore and Chung-Kil Hur</i>               |     |
| Categorical Views on Computations on Trees . . . . . | 619 |
| <i>Ichiro Hasuo, Bart Jacobs, and Tarmo Uustalu</i>  |     |

**Session A13**

|  |     |
|--|-----|
| Holographic Algorithms: The Power of Dimensionality Resolved . . . . . | 631 |
| <i>Jin-Yi Cai and Pinyan Lu</i>  |     |
| Reconciling Data Compression and Kolmogorov Complexity . . . . .       | 643 |
| <i>Laurent Bienvenu and Wolfgang Merkle</i>                            |     |
| Size Competitive Meshing Without Large Angles . . . . .                | 655 |
| <i>Gary L. Miller, Todd Phillips, and Donald Sheehy</i>                |     |

**Session B4**

|  |     |
|--|-----|
| A Fully Abstract Trace Semantics for General References . . . . .                  | 667 |
| <i>James Laird</i>   |     |
| Aliased Register Allocation for Straight-Line Programs Is<br>NP-Complete . . . . . | 680 |
| <i>Jonathan K. Lee, Jens Palsberg, and<br/>Fernando Magno Quintão Pereira</i>      |     |

Conservative Ambiguity Detection in Context-Free Grammars . . . . . 692  
*Sylvain Schmitz*

**Session A14**

Lower Bounds for Quantile Estimation in Random-Order and  
 Multi-pass Streaming . . . . . 704  
*Sudipto Guha and Andrew McGregor*

Streaming and Fully Dynamic Centralized Algorithms for Constructing  
 and Maintaining Sparse Spanners . . . . . 716  
*Michael Elkin*

Checking and Spot-Checking the Correctness of Priority Queues . . . . . 728  
*Matthew Chu, Sampath Kannan, and Andrew McGregor*

**Session B5**

Undecidability of 2-Label BPP Equivalences and Behavioral Type  
 Systems for the  $\pi$ -Calculus . . . . . 740  
*Naoki Kobayashi and Takashi Suto*

Ready Simulation for Concurrency: It’s Logical! . . . . . 752  
*Gerald Lüttgen and Walter Vogler*

Continuous Capacities on Continuous State Spaces . . . . . 764  
*Jean Goubault-Larrecq*

**Session A15**

On the Chromatic Number of Random Graphs . . . . . 777  
*Amin Coja-Oghlan, Konstantinos Panagiotou, and Angelika Steger*

Quasi-randomness and Algorithmic Regularity for Graphs with General  
 Degree Distributions . . . . . 789  
*Noga Alon, Amin Coja-Oghlan, Hiệp Hàn, Mihyun Kang,  
 Vojtěch Rödl, and Mathias Schacht*

Complexity of the Cover Polynomial . . . . . 801  
*Markus Bläser and Holger Dell*

**Session B6**

A Generalization of Cobham’s Theorem to Automata over Real  
 Numbers . . . . . 813  
*Bernard Boigelot and Julien Brusten*



|   |     |
|---|-----|
| Minimum-Time Reachability in Timed Games .....  | 825 |
| <i>Thomas Brihaye, Thomas A. Henzinger, Vinayak S. Prabhu, and Jean-François Raskin</i> |     |
| Reachability-Time Games on Timed Automata .....   | 838 |
| <i>Marcin Jurdziński and Ashutosh Trivedi</i>   |     |
| Perfect Information Stochastic Priority Games .....                                     | 850 |
| <i>Hugo Gimbert and Wiesław Zielonka</i>  |     |
| <b>Session B7</b>   |     |
| Bounded Depth Data Trees .....  | 862 |
| <i>Henrik Björklund and Mikołaj Bojańczyk</i>   |     |
| Unranked Tree Automata with Sibling Equalities and Disequalities .....                  | 875 |
| <i>Wong Karianto and Christof Löding</i>  |     |
| Regular Languages of Nested Words: Fixed Points, Automata, and Synchronization .....    | 888 |
| <i>Marcelo Arenas, Pablo Barceló, and Leonid Libkin</i>                                 |     |
| A Combinatorial Theorem for Trees .....   | 901 |
| <i>Thomas Colcombet</i>   |     |
| <b>Session B8</b>   |     |
| Model Theory Makes Formulas Large .....   | 913 |
| <i>Anuj Dawar, Martin Grohe, Stephan Kreutzer, and Nicole Schweikardt</i>               |     |
| Decision Problems for Lower/Upper Bound Parametric Timed Automata .....                 | 925 |
| <i>Laura Bozzelli and Salvatore La Torre</i>  |     |
| On the Complexity of LTL Model-Checking of Recursive State Machines .....               | 937 |
| <i>Salvatore La Torre and Gennaro Parlato</i>   |     |
| <b>Paper Retraction</b>   |     |
| Paper Retraction: On the Hardness of Embeddings Between Two Finite Metrics .....        | 949 |
| <i>Matthew Cary, Atri Rudra, and Ashish Sabharwal</i>                                   |     |
| <b>Author Index</b> .....   | 951 |

# Ushering in a New Era of Algorithm Design

Bernard Chazelle

Department of Computer Science, Princeton University, Princeton, NJ 08540, USA

**Abstract.** Advances in data acquisition technology, together with the imminent demise of Moore's Law, are prompting a rethink of basic algorithm design principles. Computing with massive data sets, data streaming, coping with uncertainty, priced computation, property testing, and sublinear algorithms are all parts of the story. So is the growing trend toward using algorithms as modeling tools for natural phenomena. I will discuss some of these developments; in particular, dimension reduction, sublinear algorithms, online reconstruction, and self-improving algorithms.

# A “proof-reading” of Some Issues in Cryptography

Ivan Damgård

Department of Computer Science, BRICS, University of Aarhus

**Abstract.** In this paper, we identify some issues in the interplay between practice and theory in cryptography, issues that have repeatedly appeared in different incarnations over the years. These issues are related to fundamental concepts in the field, e.g., to what extent we can prove that a system is secure and what theoretic results on security mean for practical applications. We argue that several such issues are often overlooked or misunderstood, and that it may be very productive if both theoreticians and practitioners think more consciously about these issues and act accordingly.

## 1 Introduction

Design of systems using cryptography is a delicate, error-prone and difficult task. Despite the fact that cryptography is probably the part of general IT security that we understand best, people frequently get it wrong. There are many examples of this, for instance the design of the encryption for the GSM mobile phone system, where parity bits for error correction were added before the encryption, making cryptanalysis much easier [4]. Or a previous version of the PKCS#1 standard for RSA encryption where it was overlooked that error messages produced by the decryption algorithm could reveal information that an adversary could exploit [2].

There are several reasons for this state of affairs. Of course, simple lack of expertise is one reason, but more fundamentally, what appears to be “common sense” and “reasonable” to the designer may only be reasonable with respect to the attacks that the designer can imagine, whereas of course an attacker in real life may well do things that are completely unexpected. Second, a system may be reasonable in the application scenario that the designer has in mind, whereas systems in reality are often applied in contexts that the designer never anticipated.

As a concrete illustration, let us assume we design a protocol for exchanging confidential data based on RSA encryption. The hope is that to break the protocol, one has to invert the RSA encryption function, i.e., given public key  $n, e$  and ciphertext  $x^e \bmod n$ , find  $x$ . But as experience shows, there is a risk that we have overlooked shortcuts that the adversary can exploit to break the system easily, without having to invert the RSA function.

A well-known and very useful way to tackle these problems is to provide a rigorous proof that if an attacker could break the cryptosystem or protocol in

question, then he could efficiently solve a computational problem that we can reasonably assume is hard (the RSA problem in our example). In this way we have assurance that no shortcuts have been overlooked in our system. This approach, which originates in the work of Goldwasser and Micali [8], is in our opinion one of the most productive and valuable contributions theoretical cryptography has to offer. Nevertheless, the actual substance of this contribution is surprisingly often misunderstood, over- or underestimated, to the extent that the author of this paper has felt it has become necessary to clarify a few issues in this context.

## 2 What Does “Provable Security” Mean?

The way to argue about security we have sketched above is often called “provable security”, but this terminology is somewhat misleading, for two reasons:

First, a proof of the form just sketched does not actually prove security. It only proves that an efficient algorithm for successfully attacking the system would imply an efficient algorithm for solving the underlying computational problem. Granted, this contradicts the assumption that the underlying problem is hard - but unfortunately, we do not know how to prove that any concrete problem really is hard to solve in the sense that would be required here. In the following, we will therefore use the term *security reduction*, since the essential part of the proofs we consider is a reduction that takes an algorithm for attacking a system and builds from it an algorithm for solving a computational problem.

A second reason is that one may get the incorrect impression that once we have “proved security”, we are home free. In reality, the situation is more complex: what a security reduction proves more precisely is that if we can break the system in time  $T$ , then we can solve the underlying problem in time  $f(T)$ , for some function  $f$ . Ideally, we want the reduction to be tight, that is,  $f(T)$  is approximately equal to  $T$ . This allows us to draw the strongest possible conclusion: if our assumption on the underlying problem says that it cannot be solved in time less than some bound  $B$ , then  $f(T)$  and hence  $T$  must be larger than  $B$ . In our RSA example, if we choose a large enough modulus so that the best known algorithms for breaking RSA would take time corresponding to, say  $2^{80}$  elementary operations on a computer, then with a tight reduction we can say that an adversary will also take time  $2^{80}$  to break our system - or he will have to find a completely new factoring algorithm. Some security reductions are not that good, we may only know that, e.g.,  $f(T) = T^2$ . This means that we need to have larger key sizes before the reduction allows us to conclude that  $T$  is sufficiently large [1]. Another issue is the exact nature of the underlying problem: if the assumption that the problem is hard is to be credible, the problem needs to be as simple, natural and well studied as possible. If we are not careful when making assumptions, we risk making useless statements that are not much better than saying “the system is secure under the assumption that it is secure”.

---

<sup>1</sup> A non-tight security reduction does not, however, mean that a larger key size is *necessary*. Maybe a tight reduction exists that we just haven’t found yet.

A final issue is that security is always proved in a specific model, where we make assumptions on what the adversary can do and what information is available to him. For instance, we may try to prove that our RSA based protocol is secure under chosen message attacks, where the adversary can submit any input he likes to the decryption algorithm and see what output is produced. The goal is then to decrypt some target ciphertext that was not decrypted “for free”. This model assumes, for instance, that the adversary has no information on the internal state of the decryption algorithm. In some cases, such information may be (partially) available, for instance if the adversary can measure radiation from the decryption equipment. This can be modeled too, using the notion of physically observable cryptography [10], for instance. But one has to be aware that some model must be chosen in order to give a proof, and a protocol may well be secure in one model and not in another.

To summarize, we have pointed out that:

- A “proof of security” never proves security in an absolute sense, it relates security to an unproven assumption that some computational problem is hard.
- The quality of a security reduction should not be ignored – it matters how tight it is, and how strong the underlying assumption is.
- A security reduction only proves something in a particular model specifying what the adversary has access to and can do.

These observations are not new at all. They have been made many times, e.g., by Bellare [1], and also in a recent paper by Kobitz and Menezes (K&M) [9]. The interesting question is, however, what conclusions we should draw? how should cryptographers and practitioners act, having understood these facts? Some researchers, including K&M, take a very critical stand on “provable security”. Provoked, perhaps, by the limitations we pointed out above, they find that the whole approach is of questionable value, and in particular researchers who let their work be guided by the goal of showing security reductions for their systems are, in fact, misguided. The argument seems to be that people build unnecessarily complicated and contrived systems only to be able to provide a security reduction for them, and also that people wrongly accept even very inefficient security reductions as an argument for believing in the security of a system.

In our opinion, such a critique misses several important points. A first high-level comment is that, while the security reduction technique does not allow us to draw simple conclusions such as “it is secure”, this does not mean that security reductions are not useful. What it does mean is that reality is more complex than we would perhaps have liked it to be, but this is not something we can blame on our scientific methods. In the following subsections, we comment in more detail on why we believe security reductions are useful despite their limitations.

## 2.1 Even Inefficient Security Reductions Are Useful

Consider again our running example, an RSA-based protocol for exchanging confidential data. Such a protocol, namely an older version of the PKCS#1 standard,

was broken under a chosen ciphertext attack by Bleichenbacher [2], as mentioned above. However, his attack did not break RSA at all, in fact the whole point of the attack was to sidestep the RSA problem and instead exploit a design failure in the protocol. Hence, Bleichenbacher’s attack is of no help at all towards inverting the RSA function. Now, if there had been a security reduction for PKCS#1, relating its security on the RSA function, this would have meant that ANY polynomial time attack would also be a polynomial time algorithm for inverting RSA, even if the reduction had not been tight. Hence any such reduction would have excluded attacks such as Bleichenbacher’s. The conclusion is that if a system has a non-tight security reduction to a well studied and standard problem, this does not help us towards choosing key sizes for the system, but it does imply that the design has no short-cuts allowing attacks that side-step the problem we were trying to base the system on. This is a very useful piece of information, also in practice: from experience, design errors usually materialize exactly as attacks that circumvent the hard problem we were trying to use.

## 2.2 Security Reductions Should Be a Design Goal

It has become standard in most cryptographic research to give security reductions for any construction that is proposed. Consequently, researchers are usually not satisfied with a construction where they cannot make a reduction go through, and will try to modify their ideas until the proof works, and often this concern influences already the early phases of a research project.

As mentioned, some researchers object to this way of working because they find that it leads to unnatural and unnecessarily complicated systems. So is it better to first build a “natural” system and then try to prove something about it? A first comment is that being “natural” is very subjective notion and hardly one that can be used a guideline for designing cryptographic constructions. Furthermore, even if one accepts that being “natural” is a possible design goal, there is no compelling reason why this would exclude the existence of a security reduction.

Granted, sometimes there seems to be a cost involved in having a security reduction – some extra ingredient that the designers add, apparently only to make the reduction go through. But on the other hand, this might just as well be exactly the ingredient that prevents a devastating attack – an attack that even the designers were not aware of! As long as no one has proved that such an attack does not exist, we simply don’t know whether the extra ingredient is superfluous or essential to security.

We believe that the only reasonable approach is to construct cryptographic systems with the objective of being able to give security reductions for them. Of course, we should not be happy about any reduction, we should strive for reductions that are efficient, and reduce to as weak assumptions as possible. And of course, we want as simple and efficient systems as possible. But on the other hand, we should not settle for protocols just because we think they “look natural” and “seem to be secure”. We return to this issue in more detail after we have looked at an example in the next section.

### 3 An Example: The Adaptive Adversary

To further illustrate the issues discussed above, we explain an example problem in more detail, and draw some conclusions for practice and theory that hopefully apply also beyond the example itself.

Consider an adversary who monitors the traffic on a network. This traffic is encrypted using public-key cryptography, we assume that each machine on the net has a private/public key pair set up, so every message is encrypted under the public key of the receiver. At the end of the day, based on what the adversary has seen, he decides to break into one of the computers on the net<sup>2</sup>. We will assume that this gives him access to the private information on that machine, including the private key, so he can now decrypt all messages intended for this machine. Based on what he now knows, he may decide on another machine to break into, etc. His resources do not, however, allow him to break into all machines. An attack as we have sketched here is called *adaptive* because the adversary decides where to break in adaptively during his attack, instead of making all decisions initially.

We will assume that the public-key cryptosystem used is secure (we discuss below what exactly this should mean) and that keys for different machines have been independently generated. If  $A$  is a subset of the machines,  $M_A$  will denote the set of all message sent to machines in  $A$ . The question now is: what can the adversary learn from such an attack? Clearly, from observing the traffic itself, he will be able to identify the sender and receiver of each message, and learn its length. But what about the information contained in the messages? If the adversary has broken into a subset  $A$  of machines, can we conclude that the adversary has learned ONLY  $M_A$ ?

It is very tempting to conclude that the answer is clearly yes: since keys are independent, seeing private keys of machines in  $A$  tells you nothing about the other private keys, and any reasonable notion of security for the cryptosystem should mean that if you do not know the private key, seeing the ciphertext tells you nothing about the plaintext.

A moment's reflection will show, however, that the question may not be quite so straightforward. If the above intuition is good, then this should mean that there is some well-defined notion of security, such that if the cryptosystem satisfies it, then the adversary can learn no extra information. But existing definitions of security of public-key encryption talk about only a single public/private key pair, where the private key is (of course) never revealed to the adversary. In our case, we have a set of many keys, where a priori *every* private key may potentially be revealed. Although only a subset is revealed in every concrete instance of the game, even the adversary may not know in advance which subset it will be (since this depends on information he will see later). Maybe this is of no real consequence, but it does demonstrate that we need a proof to be sure about our

---

<sup>2</sup> We assume for simplicity that first, all messages are sent and then the adversary starts breaking into machines. One may of course consider more general models, but this makes no essential difference to our discussion.

conclusion – at least if we want to base ourselves on a simple assumption that we can understand and have confidence in.

One way to approach the problem is to note that ideally, we want that the adversary learns only the plaintexts he can trivially decrypt. In other words, it should be as if he has access to an oracle  $O$  telling him those plaintexts. More precisely, we define  $O$  such that on input the name of a machine  $n$  in the network, it will return  $M_{\{n\}}$ . If we can show that in a real attack, the adversary learns nothing more than he could learn from sending the names of corrupted machines to  $O$ , this will certainly be sufficient. This can be done by building a simulator  $S$ , which on one side talks to  $O$  and on the other side to the adversary. Whenever the adversary wants to break into a machine, we allow  $S$  to send the name of this machine to  $O$  and get its messages. Towards the adversary,  $S$  must simulate everything the adversary would see in a real attack, including all the ciphertexts, in such a way the adversary cannot distinguish this from a real attack. If this is the case, we have certainly demonstrated that the adversary learns no extra information: everything he sees, in particular the ciphertexts, can be convincingly simulated based only on the information he was supposed to learn.

Unfortunately, an attempt to build such a simulator immediately runs into a problem:  $S$  must initially simulate the ciphertexts that  $A$  expects to see. At this point,  $S$  does not know any plaintexts: it is not allowed to access  $O$  before the adversary breaks into a machine. It therefore seems it can do nothing but create some ciphertext on its own. Let one of these ciphertexts be  $c_i = E_{pk_i}(m)$ , where we assume it is intended for machine number  $i$ , with public key  $pk_i$ .  $S$  has created  $c_i$  as an encryption of message  $m$ . If the adversary later breaks into machine  $i$ ,  $S$  will learn from  $O$  a message  $m_0$ , the message that was “really” sent. But now it is too late! When the adversary decrypts  $c_i$ , he will get  $m$  as result and not  $m_0$ . This simulation strategy therefore does not work: what the adversary sees is clearly different from the real view. No simple way to fix the problem is known. For instance, it is completely unreasonable to hope that  $S$  can guess all plaintexts correctly ahead of time.

There is a solution, however, which was found by observing that the problem comes from the fact that with standard public key encryption,  $S$  effectively commits to a decryption result by giving a ciphertext to the adversary. The idea is therefore to build a so-called *non-committing* encryption scheme. Here, we design the encryption process such that  $S$  can create a “fake” ciphertext  $c_i$  that looks like a real one, but does not contain any particular plaintext. When later  $S$  is given the right message  $m_0$ , it can create secret information related to  $c_i$ , so it looks as if  $m_0$  was in fact the encrypted plaintext. This exactly solves the problem and allows the simulation to go through.

The known implementations of non-committing encryption require interaction between sender and receiver and generation of new key material for every message sent. They are therefore much less efficient than standard encryption schemes. It can even be shown that the interaction is unavoidable: no non-interactive, non-committing encryption scheme exists [11].



So are the extra complications involved in using non-committing encryption really necessary? There are two possibilities:

- The initial intuition about the system is correct, and the adversary really cannot learn extra information, even if standard public-key cryptography is used. This may be the case - the fact that the first attempt at a simulation did not work only shows that this particular proof technique fails, not that the result is false. Maybe there is a different simulation strategy, or a proof that is not based on simulation at all?
- The initial intuition is wrong, and there is in fact an attack which can only be prevented if we use non-committing encryption. This too is entirely possible: knowing the history of the field, one would have to be very naive to believe that because we cannot think of an attack, no attack exists.

What should theoreticians do in such a situation? Skeptics of provable security may tend to think that non-committing encryption is just an unnecessary complication and that standard encryption is of course good enough. On the other hand, some theoreticians may conclude that “the book is now closed” because non-committing encryption solves the problem. In our opinion, both attitudes are wrong: there is still an open problem left! Namely, show that non-committing encryption is necessary or that standard encryption suffices. This is all the more fascinating as a basic research problem because the intuition we have not been able to prove seems so compelling.

What about the practical side? Many practitioners would probably tend to trust the intuition and harvest the efficiency advantage in using standard public-key encryption. Indeed, this is what is happening in practice. It is very important to emphasize that there is nothing “wrong” or “illegal” about this, even from a theoretical point of view. *Not as long as you understand the extra risk that is being taken*, namely that the second possibility above turns out to be the truth. After all, any practical application takes risks, for instance the risk that the underlying computational assumption is false. However hard it may be, one always has to estimate the risk taken and balance it against the practical advantages offered by the design. And it is important that theoreticians help as much as possible to identify and estimate the risks, even if it means making guesses beyond what we know for sure.

### 3.1 Do Random Oracles Help?

There is a somewhat surprising connection between the example just discussed and the random oracle model. This also deserves a discussion here, since the random oracle model has been the subject of much debate and is very central in the interplay between theory and practice.

In this model one assumes that all players have oracle access to a random function  $R$ . This means that any player can submit an input  $x$  to the oracle and will get  $R(x)$  back. Moreover, the only way to get an output is to explicitly access the oracle and no one has any information about the value  $R(x)$  until after someone has sent  $x$  to the oracle.

This model was introduced by Bellare and Rogaway [3] in order to formalize the intuition behind various applications of hash functions. When we use a hash function  $H$ , for instance as proposed by Fiat and Shamir [7] to build a signature scheme from an interactive protocol, there seems to be no way that an adversary could exploit the internal structure of  $H$  to attack the system. More concretely, if  $H$  is sufficiently complex, it seems that the attacker would have to treat  $H$  as if it was a random oracle. If this really is the case, we may as well assume that the system is actually run in the random oracle model, i.e., we may replace  $H$  by  $R$ .

In the random oracle model, one can prove that the Fiat-Shamir construction and many other constructions are secure. Unfortunately, this does not imply that these systems are secure in the real world. At most, it guarantees security from a certain class of attacks that treat the hash function as a black-box. But you do not get security in general because a concrete hash function is of course not a random oracle: as soon as you have specified the function, all outputs are fixed and predictable.

It is known that a security reduction in the random oracle model is not always useful. Several results have demonstrated that there exist systems which are secure in the random oracle model, but become insecure as soon as we replace  $R$  by a concrete function, no matter which function we use – see, e.g., [6,5]. On the other hand, these examples have been specifically engineered to demonstrate the difficulty of instantiating the random oracle by a specific function: the system tries to detect whether it is in the random oracle model or in the real world (by interacting with  $R$  or  $H$ ) and “on purpose” does something that is insecure if it finds it is in the real world.

These results therefore do not show that the Fiat-Shamir construction is insecure, for instance. Indeed one might still claim that if we use our hash function in a “sensible” way that is not designed to break down in the real world, everything should be fine, and we should be happy about a security proof in the random oracle model.

However, an observation by Jesper Buus Nielsen [11] shows that there is good reason to be skeptical about this point of view. He shows an extremely simple way to build a non-committing encryption scheme based on a random oracle: Suppose we have a public/secret key pair  $(pk, sk)$  for a cryptosystem that is secure in the standard sense. Then, to encrypt message  $m$ , we choose a random value  $r$  and the ciphertext is the pair  $(E_{pk}(r), R(r) \oplus m)$ . To decrypt using  $sk$ , one decrypts the first part to get  $r$ , calls the oracle to get  $R(r)$  and xor’s with the last part to get  $m$ .

In the random oracle model, this is a non-committing encryption scheme in the sense described above: in the simulation proof from the previous section, whenever the simulator  $S$  needs to show a ciphertext to the adversary, it can do the following: choose values  $r, s$  at random and let the “ciphertext” be  $(E_{pk}(r), s)$ . Note that  $S$  has no particular message in mind at this point, but nevertheless what is produced looks exactly like a valid ciphertext. When later  $S$  wants

to claim that some particular message  $m_0$  was encrypted, it uses the fact that because  $r$  was random and has been encrypted under a secure encryption scheme, the adversary does not know  $r$ . And so with overwhelming probability, no one has yet called the oracle with input  $r$ . Therefore,  $R(r)$  is undefined at this point, so  $S$  is free to define that  $R(r) = m_0 \oplus s$ . This is indeed a random value, and exactly makes it seem as if  $m_0$  was the encrypted message.

But note that this scheme is non-interactive! And as mentioned above, no non-interactive, non-committing encryption scheme exists *in the real world*, where random random oracles are not available. This immediately implies that no matter which concrete function we use in place of  $R$ , the resulting scheme does not work anymore. Therefore, this construction is only secure in the random oracle model, there is no way we can use it in a real application.

What does this tell us? One can hardly claim that the scheme we have just seen was designed to make the random oracle model look bad. In fact it looks quite natural, at least at first sight. The lesson to learn is that deciding whether a scheme in the random oracle model has a chance of being secure in real life is not a trivial problem. It also emphasizes that, while having a security reduction in the random oracle model is better than nothing and can be useful as a sanity check, having one in the real world is highly preferable. It is therefore a well justified research goal to build constructions that work without random oracles, but are as efficient as those that need them.

## 4 Conclusion

Many cryptographic constructions that are used in practice have a theoretical justification that leaves something to be desired. Perhaps there is no security reduction, or the one we have is not tight, or it only works in the random oracle model. Any such application takes a risk, beyond the obvious one, that the basic cryptography may not be secure. Theoreticians should not dismiss such applications as “bad practice”, but should instead make an effort to try to help estimate or reduce the risk that is being taken. This can be done by improving the constructions and/or the security reductions, but also by coming up with completely new solutions. Even if a theoretically nice but impractical solutions exists, this is not a good reason to stop working on the problem.

On the other hand, it is equally important that practitioners begin to understand that the theory of cryptography cannot always deliver simple answers such as “it is secure” or “it isn’t”. And that hence using cryptography as a black box, with no knowledge at all about what is inside, may be convenient but also potentially harmful to the security of your system. And finally, that our knowledge about the security of even well known schemes is not static, but develops over time, and this may require changes to practice that may not be convenient, but nevertheless necessary to decrease the security risks you are taking.

## References

1. Bellare, M.: Practice-oriented provable-security. In: Proceedings of First International Workshop on Information Security (ISW 97), pp. 221–231. Springer, Heidelberg (1997)
2. Bleichenbacher, D.: Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
3. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
4. Barkan, E., Biham, E., Keller, N.: Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 600–616. Springer, Heidelberg (2003)
5. Bellare, M., Boldyreva, A., Palacio, A.: An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer, Heidelberg (2004)
6. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Model Revisited. In: Proceedings of STOC (1998)
7. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
8. Goldwasser, S., Micali, S.: Probabilistic Encryption. *J. Comput. Syst. Sci.* 28(2), 270–299 (1984)
9. Koblitz, N., Menezes, A.: Another look at “Provable Security”. *J.Cryptology* 20(1), 3–37 (2007)
10. Micali, S., Reyzin, L.: Physically Observable Cryptography (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
11. Nielsen, J.B.: Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)

# Credentials-Based Authorization: Evaluation and Implementation

## Abstract of Plenary Lecture

Fred B. Schneider\*

Department of Computer Science  
Cornell University  
Ithaca, New York 14858  
U.S.A  
`fbs@cs.cornell.edu`

Nexus is a new operating system that runs on computers equipped with tamper-proof secure co-processors; it is designed to support the construction of *trust-worthy applications*—applications where actions can be attributed with some measure of confidence and where trust assumptions are explicit. To realize these goals, Nexus implements

- a novel architecture, which enables the trusted computing base to be relatively small,
- an expressive and flexible framework for making and using attestations about current and future states of a computation, and
- a high-performance cryptographically-implemented memory-isolation abstraction.

This talk focuses on how authorization policies are specified and implemented in Nexus.

We posit a guard for each resource, as has become standard. That guard receives client requests for access to the resource; it either grants or denies each request. Nexus innovates in how guards make authorization decisions and in what information is used. Our approach builds on logics having “says” and “speaks for” operators, and some surprising technical issues arise (some of which present opportunities for future research).

In Nexus, authorization decisions can involve a set of credentials that either accompany the request or that the guard obtains from elsewhere. Each *credential* is a statement whose validity can be checked by any principal. An authorization policy defines a set of requirements satisfied by suitable credentials. Given, for example, the policy that some file is accessible only to students, a guard would authorize a read request from Alice only if that guard obtains credentials attesting to Alice being a student: an attribute certificate about Alice’s student status this semester, signed by a key purporting to speak for a dean; a delegation certificate asserting that the dean’s key speaks for the dean; a delegation certificate

---

\* Supported in part by AFOSR grant F9550-06-0019, National Science Foundation Grants 0430161 and CCF-0424422 (TRUST), and a gift from Microsoft Corporation.

signed by the university president’s key asserting that the dean is indeed the holder of that office; and so on.

The advantage of such *credentials-based authorization* is that it can decentralize authorization decisions in a way that mirrors an actual authority structure. Different principals are trusted on different aspects of the over-all authorization decision, according to their expertise or access to appropriate information. Moreover, the existence of suitable credentials at a guard documents the role participating principals played in determining each authorization decision outcome and, therefore, provides an auditable justification for that decision.

In Nexus, as in prior work, every authorization policy is encoded as a formula, herein called the *authorization goal formula*, in a logic. Credentials are checked for validity, and the (sub)set of valid credentials are treated as axioms of a logic. In the prior work, however, access requests are allowed if and only if the guard can establish that the authorization goal formula for that request is valid in all models satisfying the axioms (viz, associated credentials). The guard thus must implement a theorem prover, a decision procedure, or—if requests must be accompanied by a proof of the authorization goal formula—a proof checker.

By requiring that an authorization goal formula be valid, guards in the prior work are limited to implementing monotonic authorization policies. This is a significant restriction. It rules out many authorization policies that depend on the system state (which is always changing and might over time change in ways that invalidate a conjunct). Authorization policies that limit the number of times a particular resource may be read are an important class of authorization policies that depend on state. Also, policies that admit revocation become difficult to specify, hence enforce.

Authorization policies in Nexus need not be monotonic and may depend on the state. Implementation realities do force us to prohibit certain non-monotonic authorization policies—in particular, those that include conjuncts asserting the absence of credentials. First, it is difficult to ascertain whether such a conjunct is true; a denial of service attack might block the guard from receiving a credential even though that credential exists. Second, it is difficult to ensure that such a conjunct remains true short of freezing activity elsewhere in the system.

Nexus guards can support non-monotonic policies because the guard merely determines whether an authorization goal formula is satisfied<sup>1</sup>. That is, guards are evaluators and not validity checkers. Theorem proving is still necessary for determining the truth-value of certain clauses, because some conjuncts (e.g., “A speaks for B”) cannot always be evaluated directly by inspecting a single credential or the system state. Nexus guards thus do have access to a proof checker and, when necessary, expect requests to be accompanied by proofs.

The implementation of guards in Nexus involves:

- a means to check whether formulas in the logic are satisfied, which leads to a (new) operational interpretation of “says” and “speaks for” operators,

---

<sup>1</sup> This raises some interesting but not insurmountable issues in connection with the usual Kripke interpretations of “says” and “speaks for” logics.

- trusted ways to make system state available to guards, including a kernel-supported introspection service that provides a window not only into each process’s memory but also into various other aspects (e.g., the presence of channels and the identities of their endpoints) of executing processes,
- ways of representing credentials and conjuncts of authorization goal formulas as executables rather than only as static certificates, and
- various protocols to freeze aspects of system operation so that an authorization goal formula that is found to be true can be forced to remain true as long as needed by the semantics of the operation whose authorization was being regulated.

These innovations, supported by examples, form the core of the talk.

*Acknowledgment.* Nexus is a collaboration involving Cornell Ph.D. students Oliver Kennedy, Alan Shieh, Kevin Walsh, and Dan Williams; postdoctoral associate Patrick Reynolds; and faculty E. Gün Sirer and Fred B. Schneider. The work on credentials-based authorization reported herein is being done jointly with Kevin Walsh and E. Gün Sirer.

# Subexponential Parameterized Algorithms

Frederic Dorn<sup>1,\*</sup>, Fedor V. Fomin<sup>1,\*</sup>, and Dimitrios M. Thilikos<sup>2,\*\*</sup>

<sup>1</sup> Department of Informatics, University of Bergen, N-5020 Bergen, Norway  
{frederic.dorn,fedor.fomin}@ii.uib.no

<sup>2</sup> Department of Mathematics, National & Capodistrian University of Athens,  
Panepistimioupolis, GR-15784, Athens, Greece  
sedthilk@math.uoa.gr

**Abstract.** We present a series of techniques for the design of subexponential parameterized algorithms for graph problems. The design of such algorithms usually consists of two main steps: first find a branch- (or tree-) decomposition of the input graph whose width is bounded by a sublinear function of the parameter and, second, use this decomposition to solve the problem in time that is single exponential to this bound. The main tool for the first step is Bidimensionality Theory. Here we present the potential, but also the boundaries, of this theory. For the second step, we describe recent techniques, associating the analysis of sub-exponential algorithms to combinatorial bounds related to Catalan numbers. As a result, we have  $2^{O(\sqrt{k})} \cdot n^{O(1)}$  time algorithms for a wide variety of parameterized problems on graphs, where  $n$  is the size of the graph and  $k$  is the parameter.

## 1 Introduction

The theory of fixed-parameter algorithms and parameterized complexity has been thoroughly developed during the last two decades; see e.g. the books [23,27,35]. Usually, parameterizing a problem on graphs is to consider its input as a pair consisting of the graph  $G$  itself and a parameter  $k$ . Typical examples of such parameters are the size of a vertex cover, the length of a path or the size of a dominating set. Roughly speaking, a parameterized problem in graphs with parameter  $k$  is *fixed parameter tractable* if there is an algorithm solving the problem in  $f(k) \cdot n^{O(1)}$  steps for some function  $f$  that depends only on the parameter.

While there is strong evidence that most of fixed-parameter algorithms cannot have running times  $2^{O(k)} \cdot n^{O(1)}$  (see [33,7,27]), for planar graphs it is possible to design subexponential parameterized algorithms with running times of the type  $2^{O(\sqrt{k})} \cdot n^{O(1)}$  (see [9,7] for further lower bounds on planar graphs). For example, PLANAR  $k$ -VERTEX COVER can be solved in  $O(2^{3.57\sqrt{k}}) + O(n)$  steps,

---

\* Supported by the Norwegian Research Council.

\*\* Supported by the Project “Capodistrias” (AII 736/24.3.2006) of the National and Capodistrian University of Athens (project code: 70/4/8757).



PLANAR  $k$ -DOMINATING SET can be solved in  $O(2^{11.98 \cdot \sqrt{k}}) + O(n^3)$  steps, and PLANAR  $k$ -LONGEST PATH can be solved in  $O(2^{10.52 \cdot \sqrt{k} \cdot n}) + O(n^3)$  steps. Similar algorithms are now known for a wide class of parameterized problems, not only for planar graphs, but also for several other sparse graph classes.

Since the first paper in this area appeared [2], the study of fast subexponential algorithms attracted a lot of attention. In fact, it not only offered a good ground for the development of parameterized algorithms, but it also prompted combinatorial results, of independent interest, on the structure of several parameters in sparse graph classes such as planar graphs [1,3,5,8,11,26,29,32,34] bounded genus graphs [12,28], graphs excluding some single-crossing graph as a minor [17], apex-minor-free graphs [10] and  $H$ -minor-free graphs [12,13,14].

We here present general approaches for obtaining subexponential parameterized algorithms (Section 2) and we reveal their relation with combinatorial results related to the Graph Minors project of Robertson and Seymour. All these algorithms exploit the structure of graph classes that exclude some graph as a minor. This was used to develop techniques such as Bidimensionality Theory (Section 3) and the use of Catalan numbers for better bounding the steps of dynamic programming when applied to minor closed graph classes (Sections 4 and 5).

## 2 Preliminaries

Given an edge  $e = \{x, y\}$  of a graph  $G$ , the graph  $G/e$  is obtained from  $G$  by contracting the edge  $e$ ; that is, to get  $G/e$  we identify the vertices  $x$  and  $y$  and remove all loops and replace all multiple edges by simple edges. A graph  $H$  obtained by a sequence of edge-contractions is said to be a *contraction* of  $G$ .  $H$  is a *minor* of  $G$  if  $H$  is a subgraph of a contraction of  $G$ . We use the notation  $H \preceq G$  (resp.  $H \preceq_c G$ ) when  $H$  is a minor (a contraction) of  $G$ . It is well known that  $H \preceq G$  or  $H \preceq_c G$  implies  $\mathbf{bw}(H) \leq \mathbf{bw}(G)$ . We say that a graph  $G$  is  *$H$ -minor-free* when it does not contain  $H$  as a minor. We also say that a graph class  $\mathcal{G}$  is  *$H$ -minor-free* (or, excludes  $H$  as a minor) when all its members are  $H$ -minor-free. E.g., the class of planar graphs is a  $K_5$ -minor-free graph class.

Let  $G$  be a graph on  $n$  vertices. A *branch decomposition*  $(T, \mu)$  of a graph  $G$  consists of an unrooted ternary tree  $T$  (i.e. all internal vertices of degree three) and a bijection  $\mu : L \rightarrow E(G)$  from the set  $L$  of leaves of  $T$  to the edge set of  $G$ . We define for every edge  $e$  of  $T$  the *middle set*  $\mathbf{mid}(e) \subseteq V(G)$  as follows: Let  $T_1$  and  $T_2$  be the two connected components of  $T \setminus \{e\}$ . Then let  $G_i$  be the graph induced by the edge set  $\{\mu(f) : f \in L \cap V(T_i)\}$  for  $i \in \{1, 2\}$ . The *middle set* is the intersection of the vertex sets of  $G_1$  and  $G_2$ , i.e.,  $\mathbf{mid}(e) := V(G_1) \cap V(G_2)$ . The *width*  $\mathbf{bw}$  of  $(T, \mu)$  is the maximum order of the middle sets over all edges of  $T$ , i.e.,  $\mathbf{bw}(T, \mu) := \max\{|\mathbf{mid}(e)| : e \in T\}$ . An optimal branch decomposition of  $G$  is defined by the tree  $T$  and the bijection  $\mu$  which give the minimum width, the *branchwidth*, denoted by  $\mathbf{bw}(G)$ .

A *parameter*  $P$  is any function mapping graphs to nonnegative integers. The *parameterized problem associated with  $P$*  asks, for some fixed  $k$ , whether  $P(G) = k$  for a given graph  $G$ . We say that a parameter  $P$  is *closed under taking of*

*minors/contractions* (or, briefly, *minor/contraction closed*) if for every graph  $H$ ,  $H \preceq G / H \preceq_c G$  implies that  $P(H) \leq P(G)$ .

Many subexponential parameterized graph algorithms [11,17,28,29,32,34] are associated with parameters  $P$  on graph classes  $\mathcal{G}$  satisfying the following two conditions for some constants  $\alpha$  and  $\beta$ :

- (A) For every graph  $G \in \mathcal{G}$ ,  $\mathbf{bw}(G) \leq \alpha \cdot \sqrt{P(G)} + O(1)$   
 (B) For every graph  $G \in \mathcal{G}$  and given a branch decomposition  $(T, \mu)$  of  $G$ , the value of  $P(G)$  can be computed in  $2^{\beta \cdot \mathbf{bw}(T, \mu)} n^{O(1)}$  steps.

Conditions (A) and (B) are essential due to the following generic result.

**Theorem 1.** *Let  $P$  be a parameter and let  $\mathcal{G}$  be a class of graphs such that (A) and (B) hold for some constants  $\alpha$  and  $\beta$  respectively. Then, given a branch decomposition  $(T, \mu)$  where  $\mathbf{bw}(T, \mu) \leq \lambda \cdot \mathbf{bw}(G)$  for a constant  $\lambda$ , the parameterized problem associated with  $P$  can be solved in  $2^{O(\sqrt{k})} n^{O(1)}$  steps.*

*Proof.* Given a branch decomposition  $(T, \mu)$  as above, one can solve the parameterized problem associated with  $P$  as follows. If  $\mathbf{bw}(T, \mu) > \lambda \cdot \alpha \cdot \sqrt{k}$ , then the answer to the associated parameterized problem with parameter  $k$  is "NO" if it is a minimization and "YES" if it is a maximization problem. Else, by (B),  $P(G)$  can be computed in  $2^{\lambda \cdot \alpha \cdot \beta \cdot \sqrt{k}} n^{O(1)}$  steps.

To apply Theorem 1, we need an algorithm that computes, in time  $t(n)$ , a branch decomposition  $(T, \mu)$  of any  $n$ -vertex graph  $G \in \mathcal{G}$  such that  $\mathbf{bw}(T, \mu) \leq \lambda \cdot \mathbf{bw}(G) + O(1)$ . Because of [38],  $t(n) = n^{O(1)}$  and  $\lambda = 1$  for planar graphs. For  $H$ -minor-free graphs (and thus, for all graph classes considered here),  $t(n) = f(|H|) \cdot n^{O(1)}$  and  $\lambda \leq f(|H|)$  for some function  $f$  depending only on the size of  $H$  (see [16,21,24]).

In this survey we discuss how

- to obtain a general scheme of proving bounds required by (A) and to extend parameterized algorithms to more general classes of graphs like graphs of bounded genus and graphs excluding a minor (Section 3);
- to improve the running times of such algorithms (Section 4), and
- to prove that the running time of many dynamic programming algorithms on planar graphs (and more general classes as well) satisfies (B) (Section 5).

The following three sample problems capture the most important properties of the investigated parameterized problems.

**$k$ -VERTEX COVER.** A *vertex cover*  $C$  of a graph is a set of vertices such that every edge of  $G$  has at least one endpoint in  $C$ . The  $k$ -VERTEX COVER problem is to decide, given a graph  $G$  and a positive integer  $k$ , whether  $G$  has a vertex cover of size  $k$ . Let us note that vertex cover is closed under taking minors, i.e. if a graph  $G$  has a vertex cover of size  $k$ , then each of its minors has a vertex cover of size at most  $k$ .

*k*-DOMINATING SET. A *dominating* set  $D$  of a graph  $G$  is a set of vertices such that every vertex outside  $D$  is adjacent to a vertex of  $D$ . The *k*-DOMINATING SET problem is to decide, given a graph  $G$  and a positive integer  $k$ , whether  $G$  has a dominating set of size  $k$ . Let us note that the dominating set is not closed under taking minors. However, it is closed under contraction of edges.

Given a branch decomposition of  $G$  of width  $\leq \ell$  both problems *k*-VERTEX COVER and *k*-DOMINATING SET can be solved in time  $2^{O(\ell)}n^{O(1)}$ . For the next problem, no such an algorithm is known.

*k*-LONGEST PATH. The *k*-LONGEST PATH problem is to decide, given a graph  $G$  and a positive integer  $k$ , whether  $G$  contains a path of length  $k$ . This problem is closed under the operation of taking minor but the best known algorithm solving this problem on a graph of branchwidth  $\leq \ell$  runs in time  $2^{O(\ell \log \ell)}n^{O(1)}$ .

### 3 Property (A) and Bidimensionality

In this section we show how to obtain subexponential parameterized algorithms in the case when condition (B) holds for general graphs. The main tool for this is Bidimensionality Theory developed in [10,12,13,15,18]. For a survey on Bidimensionality Theory see [14].

**Planar graphs.** While the results of this subsection can be extended to wider graph classes, we start from planar graphs where the general ideas are easier to explain. The following theorem is the main ingredient for proving condition (A).

**Theorem 2** ([37]). *Let  $\ell \geq 1$  be an integer. Every planar graph of branchwidth  $\geq \ell$  contains an  $(\ell/4 \times \ell/4)$ -grid as a minor.*

We start with PLANAR *k*-VERTEX COVER as an example. Let  $G$  be a planar graph of branchwidth  $\geq \ell$ . Observe that given a  $(r \times r)$ -grid  $H$ , the size of a vertex cover in  $H$  is at least  $\lfloor r/2 \rfloor \cdot r$  (because of the existence of a matching of size  $\lfloor r/2 \rfloor \cdot r$  in  $H$ ). By Theorem 2, we have that  $G$  contains an  $(\ell/4 \times \ell/4)$ -grid as a minor. The size of any vertex cover of this grid is at least  $\ell^2/32$ . As such a grid is a minor of  $G$ , property (A) holds for  $\alpha = 4\sqrt{2}$ .

For the PLANAR *k*-DOMINATING SET problem, the arguments used above to prove (A) for PLANAR *k*-VERTEX COVER do not work. Since the problem is not minor-closed, we cannot use Theorem 2 as above. However, since the parameter is closed under edge contractions, we can use a *partially triangulated  $(r \times r)$ -grid* which is any planar graph obtained from the  $(r \times r)$ -grid by adding some edges. For every partially triangulated  $(r \times r)$ -grid  $H$ , the size of a dominating set in  $H$  is at least  $\frac{(r-2)^2}{9}$  (every ‘‘inner’’ vertex of  $H$  has a closed neighborhood of at most 9 vertices). Theorem 2 implies that a planar graph  $G$  of branchwidth  $\geq \ell$  can be contracted to a partially triangulated  $(\ell/4 \times \ell/4)$ -grid which yields that PLANAR *k*-DOMINATING SET also satisfies (A) for  $\alpha = 12$ .

These two examples induce the following idea: if the graph parameter is closed under taking minors or contractions, the only thing needed for the proof of (A) is to understand how this parameter behaves on a (partially triangulated) grid. This brings us to the following definition.

**Definition 1** ([12]). A parameter  $P$  is minor bidimensional with density  $\delta$  if

1.  $P$  is closed under taking of minors, and
2. for the  $(r \times r)$ -grid  $R$ ,  $P(R) = (\delta r)^2 + o((\delta r)^2)$ .

A parameter  $P$  is called contraction bidimensional with density  $\delta$  if

1.  $P$  is closed under contractions,
2. for any partially triangulated  $(r \times r)$ -grid  $R$ ,  $P(R) = (\delta_{Rr})^2 + o((\delta_{Rr})^2)$ , and
3.  $\delta$  is the smallest  $\delta_R$  among all paritally triangulated  $(r \times r)$ -grids.

In either case,  $P$  is called bidimensional. The density  $\delta$  of  $P$  is the minimum of the two possible densities (when both definitions are applicable),  $0 < \delta \leq 1$ .

Intuitively, a parameter is bidimensional if its value depends on the area of a grid and not on its width or height. By Theorem 2, we have the following.

**Lemma 1.** *If  $P$  is a bidimensional parameter with density  $\delta$  then  $P$  satisfies property (A) for  $\alpha = 4/\delta$ , on planar graphs.*

Many parameters are bidimensional. Some of them, like the number of vertices or the number of edges, are not so much interesting from an algorithmic point of view. Of course the already mentioned parameter *vertex cover* (*dominating set*) is minor (contraction) bidimensional (with densities  $1/\sqrt{2}$  for *vertex cover* and  $1/9$  for *dominating set*). Other examples of bidimensional parameters are *feedback vertex set* with density  $\delta \in [1/2, 1/\sqrt{2}]$ , *minimum maximal matching* with density  $\delta \in [1/\sqrt{8}, 1/\sqrt{2}]$  and *longest path* with density 1.

By Lemma 1, Theorem 1 holds for every bidimensional parameter satisfying (B). Also, Theorem 1 can be applied not only to bidimensional parameters but to parameters that are bounded by bidimensional parameters. For example, the *clique-transversal number* of a graph  $G$  is the minimum number of vertices intersecting every maximal clique of  $G$ . This parameter is not contraction-closed because an edge contraction may create a new maximal clique and cause the clique-transversal number to increase. On the other hand, it is easy to see that this graph parameter always exceeds the size of a minimum dominating set which yields (A) for this parameter.

**Non-planar extensions and limitations.** One of the natural approaches of extending Lemma 1 from planar graphs to more general classes of graphs is via extending of Theorem 2. To do this we have to treat separately minor closed and contraction closed parameters.

The following extension of Theorem 2 holds for bounded genus graphs:

**Theorem 3** ([12]). *If  $G$  is a graph of Euler genus at most  $\gamma$  with branchwidth more than  $r$ , then  $G$  contains a  $(r/4(\gamma + 1) \times r/4(\gamma + 1))$ -grid as a minor.*

Working analogously to the planar case, Theorem 3 implies the following.

**Lemma 2.** *Let  $P$  be a minor bidimensional parameter with density  $\delta$ . Then for any graph  $G$  of Euler genus at most  $\gamma$ , property (A) holds for  $\alpha = 4(\gamma + 1)/\delta$ .*

Next step is to consider graphs excluding a fixed graph  $H$  as a minor. The proof extends Theorem 3 by making (nontrivial) use of the structural characterization of  $H$ -minor-free graphs by Robertson and Seymour in [36].

**Theorem 4 ([13]).** *If  $G$  is an  $H$ -minor-free graph with branchwidth more than  $r$ , then  $G$  has the  $(\Omega(r) \times \Omega(r))$ -grid as a minor (the hidden constants in the  $\Omega$  notation depend only on the size of  $H$ ).*

As before, Theorem 3 implies property (A) for all minor bidimensional parameters for some  $\alpha$  depending only on the excluded minor  $H$ .

For contraction-closed parameters, the landscape is different. In fact, each possible extension of Lemma 2 requires a stronger version of bidimensionality. For this, we can use the notion of a  $(r, q)$ -gridoid that is obtained from a partially triangulated  $(r \times r)$ -grid by adding at most  $q$  edges. (Note that every  $(r, q)$ -gridoid has genus  $\leq q$ .) The following extends Theorem 2 for graphs of bounded genus.

**Theorem 5 ([12]).** *If a graph  $G$  of Euler genus at most  $\gamma$  excludes all  $(k - 12\gamma, \gamma)$ -gridoids as contractions, for some  $k \geq 12\gamma$ , then  $G$  has branchwidth at most  $4k(\gamma + 1)$ .*

A parameter is *genus-contraction bidimensional* if *a*) it is contraction closed and *b*) its value on every  $(r, O(1))$ -gridoid is  $\Omega(r^2)$  (here the hidden constants in the big- $O$  and the big- $\Omega$  notations depend only on the Euler genus). Then Theorem 5 implies property (A) for all genus-contraction bidimensional parameters for some constant that depends only on the Euler genus.

An *apex graph* is a graph obtained from a planar graph  $G$  by adding a vertex and making it adjacent to some vertices of  $G$ . A graph class is *apex-minor-free* if it does not contain a graph with some fixed apex graph as a minor. An  $(r, s)$ -*augmented grid* is an  $(r \times r)$ -grid with some additional edges such that each vertex is attached to at most  $s$  vertices that in the original grid had degree 4. We say that a contraction closed parameter  $P$  is *apex-contraction bidimensional* if *a*) it is closed under taking of contractions and *b*) its value on every  $(r, O(1))$ -augmented grid is  $\Omega(r^2)$  (here the hidden constants in the big- $O$  and the big- $\Omega$  notations depend only on the excluded apex graph). According to [10] and [13], every apex-contraction bidimensional parameter satisfies property (A) for some constant that depends only on the excluded apex graph.

A natural question appears: until what point property (A) can be satisfied for contraction-closed parameters (assuming a suitable concept of bidimensionality)? As it was observed in [10], for some contraction-closed parameters, like dominating set, the branchwidth of an apex graph cannot be bounded by any function of their values. Consequently, apex-free graph classes draw a natural combinatorial limit on the the above framework of obtaining subexponential parameterized algorithms for contraction-closed parameters. (On the other side, this is not the case for minor-closed parameters as indicated by Theorem 4.) However, it is still possible to cross the frontier of apex-minor-free graphs for the dominating set problem and some of its variants where subexponential parameterized algorithms exist, even for  $H$ -minor-free graphs, as shown in [12].

These algorithms are based on a combination of dynamic programming and the structural characterization of  $H$ -minor-free graphs from [36].

## 4 Further Optimizations

In this section, we present several techniques for accelerating the algorithms emerging by the framework of Theorem 1.

**Making algorithms faster.** While proving properties (A) and (B), it is natural to ask for the best possible constants  $\alpha$  and  $\beta$ , as this directly implies an exponential speed-up of the corresponding algorithms. While, Bidimensionality Theory provides some general estimation of  $\alpha$ , in some cases, deep understanding of the parameter behavior can lead to much better constants in (A). For example, it holds that for PLANAR  $k$ -VERTEX COVER,  $\alpha \leq 3$  (see [30]) and for PLANAR  $k$ -DOMINATING SET,  $\alpha \leq 6.364$  (see [29]). (Both bounds are based on the fact that planar graphs with  $n$  vertices have branchwidth at most  $\sqrt{4.5}\sqrt{n}$ , see [30].) Similar results hold also for bounded genus graphs [28].

On the other hand, there are several ways to obtain faster dynamic programming algorithms and to obtain better bounds for  $\beta$  in (B). A typical approach to compute a solution of size  $k$  works as follows:

- Root the branch decomposition  $(T, \mu)$  of graph  $G$  picking any of the vertices of its tree and apply dynamic programming on the middle sets, bottom up, from the leaves towards the root.
- Each middle set  $\mathbf{mid}(e)$  of  $(T, \mu)$  represents the subgraph  $G_e$  of  $G$  induced by the leaves below. Recall that the vertices of  $\mathbf{mid}(e)$  are separators of  $G$ .
- In each step of the dynamic programming, all optimal solutions for a subproblem in  $G_e$  are computed, subject to all possibilities of how  $\mathbf{mid}(e)$  contributes to an overall solution for  $G$ . E.g., for VERTEX COVER, there are up to  $2^{\mathbf{bw}(T, \mu)}$  subsets of  $\mathbf{mid}(e)$  that may constitute a vertex cover of  $G$ . Each subset is associated with an optimal solution in  $G_e$  with respect to this subset.
- The partial solutions of a middle set are computed using those of the already processed middle sets of the children and stored in an appropriate data structure.
- An optimal solution is computed at the root of  $T$ .

Encoding the middle sets in a refined way, may speed up the processing time significantly. Though, the same time is needed to scan all solutions assigned to a  $\mathbf{mid}(e)$  after introducing vertex states, there are some methods to accelerate the update of the solutions of two middle sets to a parent middle set:

*Using the right data structure:* storing the solutions in a sorted list compensates the time consuming search for compatible solutions and allows a fast computing of the new solution. E.g., for  $k$ -VERTEX COVER, the time to process two middle sets is reduced from  $O(2^{3 \cdot \mathbf{bw}(T, \mu)})$  (for each subset of the parent middle set, all pairs of solutions of the two children are computed) to  $O(2^{1.5 \cdot \mathbf{bw}(T, \mu)})$ . In [19] matrices are used as a data structure for dynamic programming that allows an updating even in time  $O(2^{\frac{\omega}{2} \mathbf{bw}(T, \mu)})$  for  $k$ -VERTEX COVER (where  $\omega$  is the fast matrix multiplication constant, actually  $\omega < 2.376$ ).

*A compact encoding:* assign as few as possible vertex states to the vertices and reduce the number of processed solutions. Alber et al. [11], using the so-called “monotonicity technique”, show that 3 vertex states are sufficient in order to encode a solution of  $k$ -DOMINATING SET. A similar approach was used in [29] to obtain, for the same problem, a  $O(3^{1.5 \cdot \mathbf{bw}(T, \mu)})$ -step updating process, that has been improved by [19] to  $O(2^{2 \cdot \mathbf{bw}(T, \mu)})$ .

*Employing graph structures:* as we will see in the Section 5, one can improve the runtime further for dynamic programming on branch decompositions whose middle sets inherit some structure of the graph. Using such techniques, the update process for PLANAR  $k$ -DOMINATING SET is done in time  $O(3^{\frac{\mathbf{bw}(T, \mu)}{2}})$  [19].

The above techniques can be used to prove the following result.

**Theorem 6** ([19]). PLANAR  $k$ -VERTEX COVER can be solved in  $O(2^{3.56\sqrt{k}}) \cdot n^{O(1)}$  runtime and PLANAR  $k$ -DOMINATING SET in  $O(2^{11.98\sqrt{k}}) \cdot n^{O(1)}$  runtime.

**Kernels.** Many of the parameterized algorithms discussed in this section can be further accelerated to time  $O(n^\theta) + 2^{O(\sqrt{k})}$  for  $\theta$  being a small integer (usually ranging from 1 to 3). This can be done using the technique of kernelization that is a polynomial step preprocessing of the initial input of the problem towards creating an equivalent one, whose size depends exclusively on the parameter. Examples of such problems are PLANAR  $k$ -DOMINATING SET [4, 8, 28],  $k$ -FEEDBACK VERTEX SET [6],  $k$ -VERTEX COVER and others [25]. See the books of [23, 27, 35] for a further reference.

## 5 Property (B) and Catalan Structures

All results of the previous sections provide subexponential parameterized algorithms when property (B) holds. However, there are many bidimensional parameters for which there is no known algorithm providing property (B) in general. The typical running times of dynamic programming algorithms for these problems are  $O(\mathbf{bw}(G)!) \cdot n^{O(1)}$ ,  $O(\mathbf{bw}(G)^{\mathbf{bw}(G)}) \cdot n^{O(1)}$ , or even  $O(2^{\mathbf{bw}(G)^2}) \cdot n^{O(1)}$ . Examples of such problems are parameterized versions of  $k$ -LONGEST PATH,  $k$ -FEEDBACK VERTEX SET,  $k$ -CONNECTED DOMINATING SET, and  $k$ -GRAPH TSP. Usually, these are problems in NP whose certificate verifications involves some connectivity question. In this section, we show that for such problems one can prove that (B) actually holds for the graph class that we are interested in. To do this, one has to make further use of the structural properties of the class (again from the Graph Minors Theory) that can vary from planar graphs to  $H$ -minor-free graphs. In other words, we use the structure of the graph class not only for proving (A) but also for proving (B).

**Planar graphs.** The following type of decomposition for planar graphs follows from a proof by Seymour and Thomas in [38] and is extremely useful for making dynamic programming on graphs of bounded branchwidth faster (see [22, 19]).

Let  $G$  be a planar graph embedded in a sphere  $\mathbb{S}$ . An  $O$ -arc is a subset of  $\mathbb{S}$  homeomorphic to a circle. An  $O$ -arc in  $\mathbb{S}$  is called a *noose* of the embedding

of  $G$  if it meets  $G$  only in vertices. The length of a noose  $O$  is the number of vertices of  $G$  it meets. Every noose  $O$  bounds two open discs  $\Delta_1, \Delta_2$  in  $\mathbb{S}$ , i.e.,  $\Delta_1 \cap \Delta_2 = \emptyset$  and  $\Delta_1 \cup \Delta_2 \cup O = \mathbb{S}$ .

We define a *sphere cut decomposition* or *sc-decomposition*  $(T, \mu, \pi)$  as a branch decomposition with the following property: for every edge  $e$  of  $T$ , there exists a noose  $O_e$  meeting every face at most once and bounding the two open discs  $\Delta_1$  and  $\Delta_2$  such that  $G_i \subseteq \Delta_i \cup O_e$ ,  $1 \leq i \leq 2$ . Thus  $O_e$  meets  $G$  only in  $\mathbf{mid}(e)$  and its length is  $|\mathbf{mid}(e)|$ . A *clockwise traversal* of  $O_e$  in the embedding of  $G$  defines the cyclic ordering  $\pi$  of  $\mathbf{mid}(e)$ . We always assume that the vertices of every middle set  $\mathbf{mid}(e) = V(G_1) \cap V(G_2)$  are enumerated according to  $\pi$ .

**Theorem 7.** *Let  $G$  be a planar graph of branchwidth at most  $\ell$  without vertices of degree one embedded on a sphere. Then there exists an sc-decomposition of  $G$  of width at most  $\ell$  that can be constructed in time  $O(n^3)$ .*

In what follows, we sketch the main idea of a  $2^{O(\mathbf{bw}(T, \mu, \pi))} n^{O(1)}$  algorithm for the  $k$ -PLANAR LONGEST PATH. One may use  $k$ -LONGEST PATH as an exemplar for other problems of the same nature.

Let  $G$  be a graph and let  $E \subseteq E(G)$  and  $S \subseteq V(G)$ . To count the number of states at each step of the dynamic programming, we should estimate the number of collections of internally vertex disjoint paths using edges from  $E$  and having their (different) endpoints in  $S$ . We use the notation  $\mathbf{P}$  to denote such a path collection and we define  $\mathbf{paths}_G(E, S)$  as the set of all such path collections. Define equivalence relation  $\sim$  on  $\mathbf{paths}_G(E, S)$ : for  $\mathbf{P}_1, \mathbf{P}_2 \in \mathbf{paths}_G(E, S)$ ,  $\mathbf{P}_1 \sim \mathbf{P}_2$  if there is a bijection between  $\mathbf{P}_1$  and  $\mathbf{P}_2$  such that bijected paths in  $\mathbf{P}_1$  and  $\mathbf{P}_2$  have the same endpoints. Denote by  $\mathbf{q-paths}_G(E, S) = |\mathbf{paths}_G(E, S) / \sim|$  the cardinality of the quotient set of  $\mathbf{paths}_G(E, S)$  by  $\sim$ .

Recall that we define  $\mathbf{q-paths}_G(E, S)$  because, while applying dynamic programming on some middle set  $\mathbf{mid}(e)$  of the branch decomposition  $(T, \mu)$ , the number of states for  $e \in E(T)$  is bounded by  $O(\mathbf{q-paths}_{G_i}(E(G_i), \mathbf{mid}(e)))$ .

Given a graph  $G$  and a branch decomposition  $(T, \mu)$  of  $G$ , we say that  $(T, \mu)$  has *Catalan structure* if for every edge  $e \in E(T)$  and any  $i \in \{1, 2\}$ ,

$$\mathbf{q-paths}_{G_i}(E(G_i), \mathbf{mid}(e)) = 2^{O(\mathbf{bw}(T, \mu))} \tag{1}$$

Now, (B) holds for planar graphs because of the following combinatorial result.

**Theorem 8 ([22]).** *Every planar graph has an optimal branch decomposition with the Catalan structure that can be constructed in polynomial time.*

The proof of Theorem 8 uses an sc-decomposition  $(T, \mu, \pi)$  (constructed using the polynomial algorithm in [38]). Let  $O_e$  be a noose meeting some middle set  $\mathbf{mid}(e)$  of  $(T, \mu, \pi)$ . Let us count in how many ways this noose can cut paths of  $G$ . Observe that each path is cut into at most  $\mathbf{bw}(T, \mu, \pi)$  parts. Each such part is itself a path whose endpoints are pairs of vertices in  $O_e$ . Notice also that, because of planarity, no two such pairs can cross. Therefore, counting the ways  $O_e$  can intersect paths of  $G$  is equivalent to counting non-crossing pairs of



vertices in a cycle (the noose) of length  $\mathbf{bw}(T, \mu, \pi)$  which, in turn, is bounded by the Catalan number of  $\mathbf{bw}(T, \mu, \pi)$  that is  $2^{O(\mathbf{bw}(T, \mu, \pi))}$ .

We just concluded that the application of dynamic programming on an sc-decomposition  $(T, \mu, \pi)$  is the  $2^{O(\mathbf{bw}(T, \mu, \pi))} n^{O(1)}$  algorithm for proving property (B) for planar graphs. By further improving the way the members of  $\mathbf{q}\text{-paths}_{G_i}(E(G_i), \mathbf{mid}(e))$  are encoded during this procedure, one can bound the hidden constants in the big- $O$  notation on the exponent of this algorithm (see [22]). For example, for PLANAR  $k$ -LONGEST PATH  $\beta \leq 2.63$ . With analogous structures and arguments it follows that for PLANAR  $k$ -GRAPH TSP  $\beta \leq 3.84$ , for PLANAR  $k$ -CONNECTED DOMINATING SET  $\beta \leq 3.82$ , for PLANAR  $k$ -FEEDBACK VERTEX SET  $\beta \leq 3.56$  [19].

In [20], all above results were generalized for graphs with bounded genus (now constants for each problem depend also on the genus). This generalization requires a suitable “bounded genus”-extension of Theorem 8 and its analogues for other problems.

**Excluding a minor.** The final step is to prove property (B) for  $H$ -minor-free graphs. For the proof of this, we need the following analogue of Theorem 8.

**Theorem 9 ([21]).** *Let  $\mathcal{G}$  be a graph class excluding some fixed graph  $H$  as a minor. Then every graph  $G \in \mathcal{G}$  with  $\mathbf{bw}(G) \leq \ell$  has an branch decomposition of width  $O(\ell)$  with the Catalan structure (here the hidden constants in the big- $O$  notations in  $O(\ell)$  and the upper bound certifying the Catalan structure in Equation (1) depend only on  $H$ ). Moreover, such a decomposition can be constructed in  $f(|H|) \cdot n^{O(1)}$  steps, where  $f$  is a function depending only on  $H$ .*

The proof of Theorem 9 is based on an algorithm constructing the claimed branch decomposition using a structural characterization of  $H$ -minor-free graphs, given in [36]. Briefly, any  $H$ -minor-free graph can be seen as the result of gluing together (identifying constant size cliques and, possibly, removing some of their edges) graphs that, after the removal of some constant number of vertices (called *apices*) can be “almost” embedded in a surface of constant genus. Here, by “almost” we mean that we permit a constant number of non-embedded parts (called *vortices*) that are “attached” around empty disks of the embedded part and have a path-like structure of constant width. The algorithm of Theorem 9, as well as the proof of its correctness, has several phases, each dealing with some level of this characterisation, where an analogue of sc-decomposition for planar graphs is used. The core of the proof is based on the fact that the structure of the embeddible parts of this characterisation (along with vortices) is “close enough” to be plane, so to roughly maintain the Catalan structure property.

Theorem 9 implies (B) for  $k$ -LONGEST PATH on  $H$ -minor-free graphs. Similar results can be obtained for all problems examined in this section on  $H$ -minor-free graphs. Since property (A) holds for minor/apex-contraction bidimensional parameters on  $H$ -minor-free/apex-minor-free graphs, we have that one can design  $2^{O(\sqrt{k})} \cdot n^{O(1)}$  step parameterized algorithms for all problems examined in this section for  $H$ -minor-free/apex-minor-free graphs (here the hidden constant in the big- $O$  notation in the exponent depend on the size on the excluded minor).

## 6 Conclusion

In Section 3, we have seen that bidimensionality can serve as a general combinatorial criterion implying property (A). Moreover, no such a characterization is known, so far, for proving property (B). In Section 5, we have presented several problems where an analogue of Theorem 9 can be proven, indicating the existence of Catalan structures in  $H$ -minor-free graphs. It would be challenging to find a classification criterion (logical or combinatorial) for the problems that are amenable to this approach.

**Acknowledgements.** We take the opportunity to warmly thank Hans Bodlaender, Erik Demaine, MohammadTaghi Hajiaghayi, Naomi Nishimura, Prabhakar Ragde and Eelko Penninx for collaborating with us on this project.

## References

1. Alber, J., Bodlaender, H.L., Fernau, H., Kloks, T., Niedermeier, R.: Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica* 33, 461–493 (2002)
2. Alber, J., Bodlaender, H.L., Fernau, H., Niedermeier, R.: Fixed parameter algorithms for planar dominating set and related problems. In: Halldórsson, M.M. (ed.) *SWAT 2000. LNCS*, vol. 1851, pp. 97–110. Springer, Heidelberg (2000)
3. Alber, J., Fan, H., Fellows, M.R., Fernau, H., Niedermeier, R., Rosamond, F.A., Stege, U.: Refined search tree technique for dominating set on planar graphs. In: Sgall, J., Pultr, A., Kolman, P. (eds.) *MFCS 2001. LNCS*, vol. 2136, pp. 111–122. Springer, Heidelberg (2001)
4. Alber, J., Fellows, M.R., Niedermeier, R.: Polynomial-time data reduction for dominating set. *Journal of the ACM* 51, 363–384 (2004)
5. Alber, J., Fernau, H., Niedermeier, R.: Parameterized complexity: exponential speed-up for planar graph problems. *J. Algorithms* 52, 26–56 (2004)
6. Bodlaender, H.: A cubic kernel for feedback vertex set. In: Thomas, W., Weil, P. (eds.) *STACS 2007. LNCS*, vol. 4393, Springer, Heidelberg (2007) (to appear)
7. Cai, L., Juedes, D.: On the existence of subexponential parameterized algorithms. *J. Comput. System Sci.* 67, 789–807 (2003)
8. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. In: Ganter, B., Godin, R. (eds.) *ICFCA 2005. LNCS (LNAI)*, vol. 3403, pp. 269–280. Springer, Heidelberg (2005)
9. [ERROR while converting LaTeX/Unicode], V.G., Klinz, B., Woeginger, G.J.: Exact algorithms for the Hamiltonian cycle problem in planar graphs. *Oper. Res. Lett.* 34, 269–274 (2006)
10. Demaine, E.D., Fomin, F.V., Hajiaghayi, M., Thilikos, D.M.: Bidimensional parameters and local treewidth. *SIAM J. Discrete Math.* 18, 501–511 (2004)
11. Demaine, E.D., Fomin, F.V., Hajiaghayi, M., Thilikos, D.M.: Fixed-parameter algorithms for  $(k,r)$ -center in planar graphs and map graphs. *ACM Trans. Algorithms* 1, 33–47 (2005)
12. Demaine, E.D., Fomin, F.V., Hajiaghayi, M., Thilikos, D.M.: Subexponential parameterized algorithms on graphs of bounded genus and  $H$ -minor-free graphs. *Journal of the ACM* 52, 866–893 (2005)

13. Demaine, E.D., Hajiaghayi, M.: Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica* (to appear)
14. Demaine, E.D., Hajiaghayi, M.: The bidimensionality theory and its algorithmic applications. *Computer Journal* (to appear)
15. Demaine, E.D., Hajiaghayi, M.: Bidimensionality: new connections between FPT algorithms and PTASs. In: SODA'05, pp. 590–601. ACM-SIAM, New York (2005)
16. Demaine, E.D., Hajiaghayi, M.T., Nishimura, N., Ragde, P., Thilikos, D.M.: Approximation algorithms for classes of graphs excluding single-crossing graphs as minors. *J. Comput. System Sci.* 69, 166–195 (2004)
17. Demaine, E.D., Hajiaghayi, M.T., Thilikos, D.M.: Exponential speedup of fixed-parameter algorithms for classes of graphs excluding single-crossing graphs as minors. *Algorithmica* 41, 245–267 (2005)
18. Demaine, E.D., Hajiaghayi, M.T., Thilikos, D.M.: The bidimensional theory of bounded-genus graphs. *SIAM J. Discrete Math.* 20, 357–371 (2006)
19. Dorn, F.: Dynamic programming and fast matrix multiplication. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 280–291. Springer, Heidelberg (2006)
20. Dorn, F., Fomin, F.V., Thilikos, D.M.: Fast subexponential algorithm for non-local problems on graphs of bounded genus. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 172–183. Springer, Heidelberg (2006)
21. Dorn, F., Fomin, F.V., Thilikos, D.M.: Catalan structures and dynamic programming on H-minor-free graphs, manuscript (2007)
22. Dorn, F., Penninx, E., Bodlaender, H., Fomin, F.V.: Efficient exact algorithms on planar graphs: Exploiting sphere cut branch decompositions. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 95–106. Springer, Heidelberg (2005)
23. Downey, R.G., Fellows, M.R.: Parameterized complexity. Springer, New York (1999)
24. Feige, U., Hajiaghayi, M., Lee, J.R.: Improved approximation algorithms for minimum-weight vertex separators. In: STOC'05, pp. 563–572. ACM Press, New York (2005)
25. Fellows, M.R.: Blow-ups, win/win's, and crown rules: Some new directions in FPT. In: Bodlaender, H.L. (ed.) WG 2003. LNCS, vol. 2880, pp. 1–12. Springer, Heidelberg (2003)
26. Fernau, H., Juedes, D.W.: A geometric approach to parameterized algorithms for domination problems on planar graphs. In: Fiala, J., Koubek, V., Kratochvíl, J. (eds.) MFCS 2004. LNCS, vol. 3153, pp. 488–499. Springer, Heidelberg (2004)
27. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2006)
28. Fomin, F.V., Thilikos, D.M.: Fast parameterized algorithms for graphs on surfaces: Linear kernel and exponential speed-up. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 581–592. Springer, Heidelberg (2004)
29. Fomin, F.V., Thilikos, D.M.: Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.* 36, 281–309 (2006)
30. Fomin, F.V., Thilikos, D.M.: New upper bounds on the decomposability of planar graphs. *Journal of Graph Theory* 51, 53–81 (2006)
31. Grohe, M.: Local tree-width, excluded minors, and approximation algorithms. *Combinatorica* 23, 613–632 (2003)
32. Gutin, G., Kloks, T., Lee, C.M., Yeo, A.: Kernels in planar digraphs. *J. Comput. System Sci.* 71, 174–184 (2005)

33. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity. *Journal of Computer and System Sciences* 63, 512–530 (2001)
34. Kanj, I., Perković, L.: Improved parameterized algorithms for planar dominating set. In: Diks, K., Rytter, W. (eds.) *MFCS 2002*. LNCS, vol. 2420, pp. 399–410. Springer, Heidelberg (2002)
35. Niedermeier, R.: *Invitation to fixed-parameter algorithms*. Oxford University Press, Oxford (2006)
36. Robertson, N.: Graph minors. XVI. Excluding a non-planar graph. *J. Combin. Theory Ser. B* 89, 43–76 (2003)
37. Robertson, N., Seymour, P.D., Thomas, R.: Quickly excluding a planar graph. *J. Combin. Theory Ser. B* 62, 323–348 (1994)
38. Seymour, P.D., Thomas, R.: Call routing and the ratcatcher. *Combinatorica* 14, 217–241 (1994)

# Competitive Algorithms for Due Date Scheduling

Nikhil Bansal<sup>1</sup>, Ho-Leung Chan<sup>2</sup>, and Kirk Pruhs<sup>2,\*</sup>

<sup>1</sup> IBM T.J. Watson Research, P.O. Box 218, Yorktown Heights, NY  
nikhil@us.ibm.com

<sup>2</sup> Computer Science Department, University of Pittsburgh  
{hlchan,kirk}@cs.pitt.edu

“As a strategic weapon, time is the equivalent of money, productivity, quality, even innovation.”

George Stalk, Boston Consulting Group

**Abstract.** We consider several online scheduling problems that arise when customers request make-to-order products from a company. At the time of the order, the company must quote a due date to the customer. To satisfy the customer, the company must produce the good by the due date. The company must have an online algorithm with two components: The first component sets the due dates, and the second component schedules the resulting jobs with the goal of meeting the due dates.

The most basic quality of service measure for a job is the *quoted lead time*, which is the difference between the due date and the release time. We first consider the basic problem of minimizing the average quoted lead time. We show that there is a  $(1 + \epsilon)$ -speed  $O(\frac{\log k}{\epsilon})$ -competitive algorithm for this problem (here  $k$  is the ratio of the maximum work of a job to the minimum work of a job), and that this algorithm is essentially optimally competitive. This result extends to the case that each job has a weight and the objective is weighted quoted lead time.

We then introduce the following general setting: there is a non-increasing profit function  $p_i(t)$  associated with each job  $J_i$ . If the customer for job  $J_i$  is quoted a due date of  $d_i$ , then the profit obtained from completing this job by its due date is  $p_i(d_i)$ . We consider the objective of maximizing profits. We show that if the company must finish each job by its due date, then there is no  $O(1)$ -speed poly-log-competitive algorithm. However, if the company can miss the due date of a job, at the cost of forgoing the profits from that job, then we show that there is a  $(1 + \epsilon)$ -speed  $O(1 + 1/\epsilon)$ -competitive algorithm, and that this algorithm is essentially optimally competitive.

## 1 Introduction

We consider several online scheduling problems that arise when customers request make-to-order products from a time-based competitive company. As an

---

\* Supported in part by NSF grants CNS-0325353, CCF-0448196, CCF-0514058 and IIS-0534531.

example, consider the Atlas Door company, whose product was industrial doors that come in a wide variety of widths, heights, and materials. Traditional companies manufactured doors in batches, and stored the resulting doors in large warehouses. Atlas built just-in-time flexible factories, investing extra money to buy tooling to reduce change-over times. Traditionally when customers order a door from the manufacturer, the customers usually had to wait a week for a response as to when their order could be filled (This wait would be even longer if the door was not in stock or scheduled for production.) Atlas invested in an efficient front-end that automated the order entry, pricing and scheduling process. Atlas could price and schedule almost all of its orders immediately. Atlas was able to charge a premium for rush orders since Atlas knew these orders could not be met by its competitors. As a result, in ten short years Atlas went from start-up company to supplying 80% of the distributors in the U.S [9]. Similar success stories for other time-based competitors, such National Bicycle and Lutron electronics, can be found in [3].

In this paper, we formulate some basic online due-date scheduling problems, and study them using worst-case competitive analysis. All of the problems that we consider share the following basic framework. Jobs arrive to the system online over time. Job  $J_i$  arrives at its release time  $r_i$ . Job  $J_i$  has a work requirement  $w_i$ , which is the time it would take to complete the job on a unit speed machine. At the time  $r_i$ , the system must quote a due date  $d_i$  for job  $J_i$  to the customer submitting the job. We assume that the scheduler may preempt jobs (and later restart them from the point of preemption). It is easy to see that preemption is necessary to avoid worst case scenarios where many short jobs arrive just after a long job has been started.

An online algorithm has to have two components: a component that sets the due date, and a component that schedules the jobs. Intuitively, setting the due dates presents a dilemma to the online algorithm. Earlier due dates, if not missed, will make the customer happier, and presumably increase profit for the company. On the other hand, setting due dates to be too early restricts the scheduler from setting earlier due dates on later arriving jobs. The main goal of this paper is to gain some understanding about when and how an online algorithm can reasonably cope with this dilemma.

The standard quality of service measure for a job  $J_i$  is the *quoted lead time (QLT)*, which is defined to be  $d_i - r_i$  [5][7]. Probably the most natural related objective is to minimize the average, or equivalently total, quoted lead times of the jobs under the constraint that all jobs must be finished by their due date. That is, the objective is  $\sum(d_i - r_i)$ . In section 2, we show that there is a  $(1 + \epsilon)$ -speed  $O(\frac{\log k}{\epsilon})$ -competitive algorithm, which we call BIT, and we show that BIT is essentially optimally competitive. More generally, we show that this result extends to the case that each job has a weight and the objective is weighted quoted lead time. The parameter  $k$  is the ratio of the maximum density of any job to the minimum density of any job, where the density of a job is its weight divided by its size. The BIT algorithm is a composition of three well known

scheduling algorithms: Highest Density First (or equivalently Shortest Job First in the unweighted setting), Round Robin, and First-Come-First-Served.

It is instructive to consider the case that the online scheduler can delay the setting of due dates. In particular, consider the most extreme case where the online scheduler can set the due date to be the completion time of a job. The quoted lead time problem then becomes the standard flow time problem, for which the online Shortest Remaining Processing Time algorithm is optimal. The weighted quoted lead time problem becomes the problem of minimizing the weighted flow time, for which the online algorithm Highest Density First is  $(1 + \epsilon)$ -speed  $O(1 + 1/\epsilon)$ -competitive. Thus, the introduction of due dates makes the resulting online problems much more difficult.

Minimizing total quoted lead times is a reasonable objective function if a company wishes to promise generally fast response to a collection of essentially equivalent customers/jobs. But in some situations, say if the company charges a premium for rush orders, the company may explicitly know how much profit it can obtain from a particular due date. For example, companies such as Atlas Door charge additional shipping and handling for rush orders. In these cases, this known profit should be incorporated into the objective function. We introduce the following general setting: there is a non-increasing profit function  $p_i(t)$  associated with each job  $J_i$ . If the customer for job  $J_i$  is quoted a due date of  $d_i$ , then the profit obtained from completing this job by its due date is  $p_i(d_i)$ . We consider the objective of maximizing profits, that is the objective is  $\sum p_i(d_i)$ . Note that the online scheduler in this problem has the power to reject jobs since setting the due date to  $+\infty$  is essentially equivalent to rejecting the job.

Consider the following dilemma for the online scheduler: after scheduling a low-value long job, several emergency jobs arrive that will yield high profit if early due dates are set. However, setting early due dates for these emergency jobs would make it infeasible to finish both the emergency jobs and the low-value job by their due dates. The company would like to be able to drop the low-value job to make greater profit from the emergency jobs. We get two different models depending on whether this dropping is allowed. In the *reliable* model, the scheduler must complete every job by its due date. In this model, the company could not get profit from these emergency jobs in our example instance. In the *unreliable* model, the company merely does not earn any profit from those jobs that it does not complete by their due dates. Many companies offer no recourse to a customer that is not provided a service by/at the time promised other than that they are not charged. If you have had a flight canceled by Ryan Air, or a package not delivered in time by Federal Express, you probably have had first hand experience with this.

Our results on profit maximization are given in section [3](#). We show that in the reliable model there is no  $O(1)$ -speed poly-log-competitive algorithm. In the unreliable model, we show that there is a  $(1 + \epsilon)$ -speed  $O(1)$ -competitive algorithm, and we show that this algorithm is essentially optimally competitive. These results match the intuition that the ability to break promises is crucial to a company that wants to maximize profit. Once again it is instructive to

consider relaxing the problem so that the online algorithm can delay the setting of the due dates until the completion time of job. In this relaxation, there is no difference between the reliable and unreliable models. A special case of this relaxed problem that has previously been studied in the literature is if the profit functions are constant until some time/deadline, and then zero thereafter. For this special case of the relaxed problem, it is known the best result that is achievable is  $O(1 + \epsilon)$ -speed  $O(1)$ -competitiveness [4]. Thus  $O(1 + \epsilon)$ -speed  $O(1)$ -competitiveness is the best possible result that we could have hoped for in our due date profit maximization problem. So for profit maximization, the introduction of due dates makes the resulting online problem significantly harder only in the reliable case.

Within the world of supply chain management there is an extensive literature on due date scheduling problems. Surveys of due date scheduling problems, with hundreds of references, can be found [5] and [7]. As always with scheduling problems, there is a vast number of reasonable formulations that have been studied. The majority of the literature uses experimentation as the method of algorithm evaluation. There is also a fair amount of literature that uses queuing-theory analysis to evaluate online algorithms.

There is to our knowledge only one previous paper, namely [6], in the literature on online due-date scheduling problems that uses worst-case analysis. In this special case, all jobs have the same work, say  $w$ . If the due date of a job is set to be as early as feasible, namely at  $r_i + w$ , then the profit obtained from this job is a fixed constant  $\ell$ . For every unit of time that the due date is delayed, one unit of profit is lost. It is easy to see that the online algorithm, that accepts a job if it can gain positive profit, has a competitive ratio of  $\Omega(\ell)$ ; This algorithm may accept many jobs with low profit, and thus not be able to reap profit from later arriving jobs. [6] shows that the reliable online algorithm that rejects jobs on which it won't earn at least 61.8% of the maximum profit  $\ell$ , is 1.618-competitive. Note that this result relies heavily on both the fact that the profit function has a very special structure, and on the fact that all jobs are identical except release times. The paper [6] considers other special cases and variations, for example, if there are two different kinds of jobs instead of one and if quotations can be delayed.

Recall that an online scheduling algorithm  $A$  is  $s$ -speed  $c$ -competitive for an objective function  $f$  if for all input instances  $f(A_s(I)) \leq c \cdot f(OPT_1(I))$ , where  $A_s(I)$  is the output of algorithm  $A$  with an  $s$  speed processor on input  $I$ , and  $OPT_1(I)$  is the optimal unit speed schedule for input  $I$  [4]. Instances that generally arise in practice for many scheduling problems have the following threshold property: The performance of the system, with an optimal scheduling algorithm, is reasonable when the load is bounded away from some threshold, but the performance is horrible when the load is above the threshold. Intuitively, a  $(1 + \epsilon)$ -speed  $c$ -competitive algorithm should perform reasonably well on such common instances since it would then have the same threshold as the optimal algorithm. For more information, see the survey [8].



## 2 Minimizing Weighted Quoted Lead Time

This section considers the problem of minimizing weighted quoted lead time. Recall that each job  $J_i$  has release time  $r_i$ , amount of work  $w_i$  and weight  $c_i$ . An online algorithm needs to set a due date  $d_i$  when  $J_i$  is released and the quoted lead time (or simply lead time) of  $J_i$  is  $\ell_i = (d_i - r_i)$ . The objective is to minimize  $\sum c_i \ell_i$ , i.e., the total weighted lead time. We define the *density* of a job  $J_i$  to be  $c_i/w_i$ . Let  $k$  be the ratio of the maximum to minimum density of the jobs. We give a simple algorithm BIT that is  $(1 + \epsilon)$ -speed  $O((\log k)/\epsilon)$ -competitive and we show that BIT already achieves a nearly optimal competitive ratio.

### 2.1 The Algorithm BIT and Its Analysis

Let us first give some motivation for BIT. For any job sequence  $I$ , let  $L$  be the minimum possible total weighted lead time and let  $F$  be the minimum possible total weighted flow time. We note that  $L = F$ , because the total weighted lead time is at least the total weighted flow time and they can be equal when the due date of each job is set to its completion time.

Consider the algorithm Highest Density First (HDF), that at any time works on the highest density job. It is known that for any  $\epsilon > 0$ , HDF is  $(1 + \frac{\epsilon}{2})$ -speed  $(1 + \frac{2}{\epsilon})$ -competitive for minimizing weighted flow time [2]. Suppose, BIT runs a copy of HDF; furthermore, whenever a job  $J_i$  is released, the due date  $d_i$  is set to at most  $\alpha$  times the completion time of  $J_i$  in HDF, assuming that no more jobs will be released. If it turned out that all jobs were satisfied by their deadlines, it would imply that BIT is  $\alpha(1 + \frac{2}{\epsilon})$ -competitive for total weighted lead time. Of course, the problem is that HDF may not complete  $J_i$  by  $d_i$  since many higher density jobs might arrive during  $[r_i, d_i]$ . Interestingly, it turns out that by giving BIT slightly extra speed, and by choosing  $\alpha$  large enough, we can guarantee that each job will be completed by its due date. We define BIT formally as follows.

We may assume without loss of generality that the minimum density of a job is 1. Increasing the weight of jobs by a factor of at most 2, we assume that all jobs have densities  $2^j$  for  $j = 1, 2, \dots, \log k$ . BIT divides the jobs into classes  $C_1, C_2, \dots, C_{\log k}$  where all jobs in  $C_j$  have density  $2^j$ . BIT operates as follows.

**Setting due dates.** When a job  $J_i$  of class  $C_j$  is released at time  $r_i$ , let  $w(C_j)$  be the amount of remaining work for jobs in class  $C_j$  at time  $r_i$ . BIT sets the due date  $d_i$  to  $r_i + (w(C_j) + w_i) \cdot \frac{2 \log k}{\epsilon}$ . Note that if no more jobs are released, a processor of speed  $\frac{\epsilon}{2 \log k}$  can complete  $J_i$  and all the jobs in  $C_j$  by  $d_i$ .

**Processing jobs.** BIT divides its processing power into two parts by time-sharing. Thus, we may assume that BIT is using a processor  $P_1$  of speed  $(1 + \frac{\epsilon}{2})$  and also another processor  $P_2$  of speed  $\frac{\epsilon}{2}$ .

- $P_1$  runs HDF, i.e., always process the highest density unfinished job.
- $P_2$  is evenly time-shared among all the  $\log k$  classes. For each class  $C_j$ , it processes the job in  $C_j$  with the earliest release time using speed  $\frac{\epsilon}{2 \log k}$ .

**Observation 1.** BIT completes each job  $J_i$  by its due date  $d_i$ .

*Proof.* Since jobs in  $C_i$  are dedicated a processor of speed  $\frac{\epsilon}{2 \log k}$ , the job  $J_i$  will be completed by  $d_i$  irrespective of the jobs that arrive after  $r_i$ .

We are now ready to bound the lead time  $\ell_i$  of each job  $J_i$  in BIT. For any  $s \geq 1$ , set  $\text{HDF}(s)$  denotes a stand-alone copy of HDF using a  $s$ -speed processor. Let  $f_i$  be the flow time of a job  $J_i$  in the schedule of  $\text{HDF}(1 + \frac{\epsilon}{2})$ . The flow time of a job is its completion time minus its release date.

**Lemma 1.** For any job  $J_i$ ,  $\ell_i \leq \frac{2 \log k}{\epsilon} \cdot (1 + \frac{\epsilon}{2}) \cdot f_i$ .

*Proof.* Let  $J_i$  be a job of class  $C_j$  and let  $w_h(C_j)$  be the amount of unfinished work under  $\text{HDF}(1 + \frac{\epsilon}{2})$  for jobs in class  $C_j$  at time  $r_i$ . Since BIT is also running a copy of  $\text{HDF}(1 + \frac{\epsilon}{2})$ , it must be that  $w(C_j)$ , the unfinished work for class  $C_j$  jobs under BIT is at most  $w_h(C_j)$ . Note that  $f_i$  is at least  $(w_h(C_j) + w_i)/(1 + \frac{\epsilon}{2})$ . Hence,  $\ell_i = (w(C_j) + w_i) \cdot \frac{2 \log k}{\epsilon} \leq (w_h(C_j) + w_i) \cdot \frac{2 \log k}{\epsilon} \leq \frac{2 \log k}{\epsilon} \cdot (1 + \frac{\epsilon}{2}) \cdot f_i$ .

**Theorem 1.** For any  $\epsilon > 0$ , BIT is  $(1 + \epsilon)$ -speed  $\frac{4 \log k}{\epsilon} (1 + \frac{\epsilon}{2})(1 + \frac{2}{\epsilon})$ -competitive. The competitive ratio can improved to  $\frac{16 \log k}{\epsilon}$  for  $\epsilon \geq 2$ .

*Proof.* Let  $Opt$  be the adversary that minimizes the total weighted lead time. By Lemma 1, we have that

$$\begin{aligned} \sum w_i \ell_i &\leq \frac{2 \log k}{\epsilon} (1 + \frac{\epsilon}{2}) \sum w_i f_i \\ &\leq \frac{2 \log k}{\epsilon} (1 + \frac{\epsilon}{2}) (1 + \frac{2}{\epsilon}) \times \text{total weighted flow time of } Opt \\ &= \frac{2 \log k}{\epsilon} (1 + \frac{\epsilon}{2}) (1 + \frac{2}{\epsilon}) \times \text{total weighted lead time of } Opt, \end{aligned}$$

which completes the first part of the proof. For  $\epsilon \geq 2$ , we note that  $\text{HDF}(1 + \frac{\epsilon}{2})$  is  $\frac{4}{1 + \epsilon/2}$ -competitive on weighted flow time (because  $\text{HDF}(1 + \frac{\epsilon}{2})$  has weighted flow time at most  $\frac{2}{1 + \epsilon/2}$  times that of  $\text{HDF}(2)$ , and  $\text{HDF}(2)$  is 2-competitive on weighted flow time). Similar as above, we conclude that for  $\epsilon \geq 2$ , the competitive ratio of BIT is  $\frac{2 \log k}{\epsilon} (1 + \frac{\epsilon}{2}) (\frac{4}{1 + \epsilon/2}) = \frac{8 \log k}{\epsilon}$ . Since the weights increased at most twice to ensure that job densities are powers of 2, the result follows.

## 2.2 Lower Bounds

**Theorem 2.** Let  $c > 1$  be any integer. Any deterministic  $c$ -speed algorithm is  $\Omega(\frac{\log k}{c})$ -competitive.

*Proof.* Consider any  $c$ -speed algorithm  $A$ . Let  $x \geq 2$  be any integer. We will release a sequence of at most  $(c \log x + 1)$  batches of jobs. All jobs have weight 1. Batch  $B_i$  has  $2^{i-1}$  jobs and each job has size  $1/2^{i-1}$ . First, batch  $B_1$  is released at time 0. For  $i = 2, \dots, c \log x + 1$ , if at least half of the jobs in  $B_{i-1}$  have due date at most  $\frac{\log x}{2}$ , then batch  $B_i$  is released immediately at time 0; otherwise,

the job sequence terminates. Note that if batch  $B_{c \log x + 1}$  is released, there must be at least  $\frac{c \log x}{2}$  units of work with due date at most  $\frac{\log x}{2}$ .  $A$  has a  $c$ -speed processor and needs to meet the due date of the jobs, so  $A$  needs to set the due date of all jobs in batch  $B_{c \log x + 1}$  to be greater than  $\frac{\log x}{2}$ .

Let  $B_r$  be the last batch of jobs released. Note that  $A$  sets the due date of at least  $\frac{2^{r-1}}{2}$  jobs to be at least  $\frac{\log x}{2}$ . The total weighted lead time of  $A$  is  $\Omega(2^{r-1} \log x)$ . The adversary can schedule the jobs in reverse order of arrival, giving an objective of at most  $\sum_{i=1}^r 2^{i-1}(r+1-i) = O(2^{r-1})$ . Thus,  $A$  is  $\Omega(\log x)$ -competitive. The theorem follows as  $k = 2^{c \log x}$  and  $\log x = \frac{\log k}{c}$ .

We remark that any deterministic algorithm using a unit-speed processor can be shown to be  $\Omega(\sqrt{k})$ -competitive.

### 3 Profit Maximization

We assume in this section that each job  $J_i$  has an associated non-increasing profit function  $p_i(t)$  specifying the profit obtained if the due date is set to time  $t$ , and our objective is the total profit obtained from the jobs finished by their due date. Our main result, which we give in section [3.1](#), is a  $(1 + 2\delta)$ -speed  $(6 + \frac{12}{\delta} + \frac{8}{\delta^2})$ -competitive algorithm in the unreliable model where the scheduler is not obligated to finish all jobs by their due date. It is easily seen (see Section [3.2](#)) that resource augmentation is necessary. In section [3.2](#) we also show that every deterministic  $c$ -speed algorithm is at least  $(1 + \frac{1}{c \cdot 2^c})$ -competitive, even when the job size and profit are bounded. In section [3.3](#), we consider the reliable version of the problem where all jobs must be completed by their due date. We show that every deterministic  $c$ -speed algorithm is  $\Omega(k^{1/c})$ -competitive, and give a simple algorithm that matches this bound.

#### 3.1 The Algorithm for the Unreliable Model

We call the algorithm INT, which stands for “intervals”. It maintains a pool  $P$  of jobs and only processes jobs in  $P$ . Whenever a job  $J_i$  is released, INT assigns a due date  $d_i$  to  $J_i$  (using the procedure below), and  $J_i$  is put into  $P$  if  $p_i(d_i) > 0$ . A job  $J_i$  remains in  $P$  until it is completed or becomes non-viable (i.e., the remaining work is more than  $(1 + 2\delta)$  times the duration between the current time and its due date). Note that once the due date of  $J_i$  is fixed, we can unambiguously define  $p_i = p_i(d_i)$  as the profit and  $u_i = p_i/w_i$  as the density of  $J_i$ .

Intuitively, INT runs the highest density job available. And intuitively INT sets the due date to be the earliest time  $t$  so that if no more jobs arrive, and always the highest density job is run, then even a slightly slower processor could finish the job by  $t$ . Unfortunately, it requires some complications to make the idea precise and to get the technical details to work out.

Let  $c > 1 + \frac{1}{\delta}$  be a constant (that we set later to  $1 + \frac{2}{\delta}$ ). Throughout, we assume that INT runs at speed  $1 + 2\delta$  and the optimum offline algorithm runs

at speed 1. We now formally specify how the due dates are set, and which job is executed at each time  $t$ .

**Setting Due Dates.** INT maintains an invariant that each job admitted in  $P$  is associated with a collection of time intervals  $I(J_i) = \{[t_1, t'_1], [t_2, t'_2], \dots, [t_h, t'_h]\}$ , where  $r_i \leq t_1 < t'_1 < t_2 < \dots < t_h < t'_h = d_i$ . This collection  $I(J_i)$  is specified (and fixed) as soon as  $J_i$  is released. The total length of the intervals in  $I(J_i)$  is  $\frac{1+\delta}{1+2\delta} w_i$ . Roughly speaking,  $J_i$  is expected to be run during  $I(J_i)$ . Note that the total length is  $(1 + \delta)$  times more than the time required to run  $J_i$ .

When a new job  $J_i$  is released at  $r_i$ , INT tests whether a time  $t' > r_i$  is a good choice for due date as follows: Assuming  $t'$  is the due date, then the profit of  $J_i$  is  $p_i(t')$  and density  $u_i = p_i(t')/w_i$ . Let  $X(\frac{u_i}{c})$  be the set of jobs in  $P$  with density at least  $\frac{u_i}{c}$ . Consider the time interval  $[r_i, t']$  and the associated intervals of jobs in  $X(\frac{u_i}{c})$ . Let  $A = \{[a_1, a'_1], [a_2, a'_2], \dots, [a_h, a'_h]\}$  be the maximal subintervals of  $[r_i, t']$  that do not overlap with the associated intervals of any job in  $X(\frac{u_i}{c})$ . We say that  $t'$  is a *feasible due date* if the total length of intervals in  $A$  is at least  $\frac{1+\delta}{1+2\delta} w_i$ . Note that a feasible due date always exists by choosing  $t'$  large enough. INT sets the due date of  $J_i$  to be the earliest time  $t'$  such that  $t'$  is a feasible due date. If  $p_i(t') > 0$ , then  $J_i$  is put into  $P$  and  $I(J_i)$  is set to the corresponding  $A$ .

**Executing jobs.** At any time  $t$ , let  $S$  be the set of jobs  $J_i$  in  $P$  that are allowed to run at  $t$  (i.e. all  $J_i$  such that  $I(J_i)$  contains some interval  $[t_j, t'_j]$  and  $t \in [t_j, t'_j]$ ). INT processes the job in  $S$  with highest density.

We first state two observations about INT before presenting the analysis.

**Observation 2.** *At any time  $t$ , let  $J_1$  and  $J_2$  be two jobs in  $P$  such that some intervals of  $I(J_1)$  and  $I(J_2)$  are overlapping. Then, either  $u_1 > c \cdot u_2$  or  $u_2 > c \cdot u_1$ .*

*Proof.* Follows directly from the procedure for defining  $I(J_1)$  and  $I(J_2)$ .

Consider the overall execution of INT in hindsight for the sequence of jobs. For any time  $t$ , let  $S$  be the set of jobs  $J_i$  ever put into  $P$  such that  $I(J_i)$  contained some interval  $[t_j, t'_j]$  and  $t \in [t_j, t'_j]$ . The density of  $t$  is defined as the density of the highest density job in  $S$ .

**Observation 3.** *Consider any job  $J_i$  and a time  $t \geq r_i + w_i$ . Suppose INT sets the due date of  $J_i$  to be strictly greater than  $t$ . Let  $u_i = p_i(t)/w_i$  and let  $L$  be the amount of time during  $[r_i, t]$  such that the density is at least  $\frac{u_i}{c}$ . Then,  $L \geq \frac{\delta}{1+2\delta}(t - r_i)$ .*

*Proof.* If  $L$  is less than  $\frac{\delta}{1+2\delta}(t - r_i)$ , then  $t$  is a feasible due date for  $J_i$  and INT would have set the due date of  $J_i$  to be at most  $t$ , which yields a contradiction.

We now turn our attention towards analyzing this algorithm. Let  $C$  be the set of jobs completed by INT, and  $R$  be the set of jobs that have ever been put into  $P$ . For any set  $X$  of jobs, let  $\|X\|$  be the total profit of jobs in  $X$  according to the due dates set by INT. We first lower bound the profit obtained by INT.

**Lemma 2.** For  $C$  and  $R$  as defined above,  $\|C\| \geq (1 - \frac{1}{\delta(c-1)})\|R\|$ , or equivalently,  $\|R\| \leq \frac{\delta(c-1)}{\delta(c-1)-1}\|C\|$ .

*Proof.* We use a charging scheme to prove the lemma. For each job  $J_i$  in  $C$ , we give  $p_i$  units of credit to  $J_i$  initially. The remaining jobs in  $R-C$  are given 0 units of credit initially. We will describe a method to transfer the credits such that at the end, each job  $J_i \in R$  has credit at least  $(1 - \frac{1}{\delta(c-1)})p_i$ , which completes the proof.

The method to transfer credit is as follows. At any time  $t$ , let  $S$  be the set of jobs that have an associated interval containing  $t$ . Let  $J_i$  be the highest density job in  $S$ . Then, for each other job  $J_j$  in  $S$ ,  $J_i$  transfers credit to  $J_j$  at a rate of  $(\frac{1+2\delta}{\delta})u_j$ .

We first show that every job  $J_j$  in  $R$  receives credit at least  $p_j$  either initially or transferred from other jobs. This clearly holds for jobs in  $C$ . For any job  $J_j$  in  $R-C$ , as  $J_j$  could not be completed during  $I(J_j)$ , it must have received credit for at least  $\frac{\delta}{1+2\delta} \cdot w_j$  units of time. Thus, the total credit obtained is at least  $(\frac{\delta}{1+2\delta})w_j \cdot (\frac{1+2\delta}{\delta})u_j = w_j u_j = p_j$ .

We now show that the credit transferred out of each job  $J_i$  is at most  $\frac{1}{\delta(c-1)}p_i$ . When a job  $J_i$  is the highest density job in  $S$ , by observation [2](#) the remaining jobs in  $S$  have geometrically decreasing densities and hence their total density is at most  $\frac{1}{c-1}u_i$ . Therefore, the rate of credit transferring out of  $J_i$  is at most  $(\frac{u_i}{c-1})(\frac{1+2\delta}{\delta})$ . Since  $J_i$  is the highest density job for at most  $\frac{w_i}{1+2\delta}$  units of time, the total credit transferred out is at most of  $J_i$  is at most  $(\frac{u_i}{c-1})(\frac{1+2\delta}{\delta}) \cdot (\frac{w_i}{1+2\delta}) = \frac{1}{\delta(c-1)}p_i$ .

Next, we upper bound the profit obtained by the adversary. Let  $A$  be the set of jobs completed by the adversary. For any set of jobs  $X$ , let  $\|X\|^*$  be the total profit of jobs in  $X$  according to the due dates set by the adversary. We may assume that the adversary only completes jobs with non-zero profit. Let  $A_1$  be the set of jobs in  $A$  such that the due date set by INT is no later than that by the adversary. Let  $A_2 = A \setminus A_1$ . Then, the total profit obtained by the adversary is  $\|A\|^* = \|A_1\|^* + \|A_2\|^* \leq \|A_1\| + \|A_2\|^* \leq \|R\| + \|A_2\|$ . Note that  $\|A_1\| \leq \|R\|$  because each job in  $A_1$  must lie in  $R$  since INT set a due date for it. We now bound the profit of jobs in  $A_2$ .

For any  $u > 0$ , let  $T(u)$  be the total length of time that the adversary is running jobs in  $A_2$  with density at least  $u$  (where the density is determined according to the due dates set by the adversary). For the schedule of INT, let  $L(\frac{u}{c})$  be the total length of time such that the density is at least  $\frac{u}{c}$ . (Recall that the density of a time  $t$  is the density of the highest density job that has an associated interval containing  $t$ .)

**Lemma 3.** For every  $u > 0$ ,  $T(u) \leq \frac{2(1+2\delta)}{\delta}L(\frac{u}{c})$ .

*Proof.* For any job  $J_i \in A_2$ , let the span of  $J_i$  be the time interval  $[r_i, d_i^*]$ , where  $d_i^*$  is the due date set by the adversary. For any  $u > 0$ , let  $A_2(u)$  be the set of

jobs in  $A_2$  with density at least  $u$ . Consider the union of spans of all jobs in  $A_2(u)$ . It may consist of a number of disjoint time intervals, and let  $\ell$  be its total length. Clearly,  $T(u) \leq \ell$ .

Let  $M \subseteq A_2$  be the minimum cardinality subset of  $A_2$  such that the union of spans of jobs in  $M$  equals that of  $A_2$ . Note that the minimality property implies no three jobs in  $M$  have their spans overlapping at a common time. This implies that we can further partition  $M$  into  $M_1$  and  $M_2$  such that within  $M_1$  (resp.  $M_2$ ), any two jobs have disjoint spans. Now, either  $M_1$  or  $M_2$  has total span of length at least half of that of  $M$ . Without loss of generality, suppose that it is  $M_1$ . Note that each interval in  $M_1$  corresponds to a span of some job in  $A_2$ . Applying Observation 3 to each such interval, it follows that the density of INT is at least  $\frac{u}{c}$  for at least  $\frac{\delta}{1+2\delta}$  fraction of time during the intervals of  $M_1$ . Thus,  $L(\frac{u}{c}) \geq \frac{\delta}{2(1+2\delta)} \cdot T(u)$ , which completes the proof.

Let  $\{\phi_1, \phi_2, \dots, \phi_m\}$  be the set of densities of jobs in  $A_2$  (determined by the adversary's due dates), where  $\phi_i > \phi_{i+1}$  for  $i = 1, \dots, m-1$ . For simplicity, let  $\phi_0 = \infty$  and  $\phi_{m+1} = 0$ . For  $i = 1, \dots, m$ , let  $\ell_i$  be the length of time that the adversary is running jobs of density  $\phi_i$ . Similarly, for  $i = 1, \dots, m$ , let  $\alpha_i$  be the length of time that INT has density in the range  $[\phi_i/c, \phi_{i-1}/c)$ . Then, the following holds.

**Lemma 4.** *Let  $K$  be a constant. If  $T(u) \leq K \cdot L(\frac{u}{c})$  for every  $u \geq 0$ , then  $\sum_{i=1}^m \ell_i \phi_i \leq K \cdot \sum_{i=1}^m \alpha_i \phi_i$ .*

*Proof.* Rephrasing  $T(i) \leq K \cdot L(\frac{i}{c})$  in terms of  $\ell_i$  and  $\alpha_i$ , we obtain the following inequalities for each  $i = 1, \dots, m$

$$\ell_1 + \dots + \ell_i \leq K(\alpha_1 + \dots + \alpha_i).$$

Multiplying the  $i^{\text{th}}$  inequality by  $(\phi_i - \phi_{i+1})$  (which is strictly positive for all  $i$ ) and adding them, we obtain the desired result that  $\sum_{i=1}^m \ell_i \phi_i \leq K \cdot (\sum_{i=1}^m \alpha_i \phi_i)$ .

**Lemma 5.**  $\|A\|^* \leq (1 + \frac{2(1+\delta)c}{\delta})\|R\|$ .

*Proof.* As  $\|A_1\| \leq \|R\|$ , it follows that  $\|A\|^* \leq \|A_1\|^* + \|A_2\|^* \leq \|R\| + \|A_2\|^*$ . By Lemma 3 and 4,

$$\|A_2\|^* \leq \sum_{i=1}^m \ell_i \phi_i \leq \frac{2(1+2\delta)}{\delta} \sum_{i=1}^m \alpha_i \phi_i. \quad (1)$$

Let  $q_i$  be the total profit for jobs whose density in INT is in the range of  $[\phi_i/c, \phi_{i-1}/c)$ . For any job  $J_j$ , as the total length of the associated intervals is  $\frac{1+\delta}{1+2\delta} w_j$ , it follows that  $\alpha_i \frac{\phi_i}{c} \leq \frac{1+\delta}{1+2\delta} \cdot q_i$ . Combining with (1), we obtain that

$$\|A_2\|^* \leq \frac{2(1+2\delta)}{\delta} \sum_{i=1}^m \alpha_i \phi_i \leq \frac{2(1+\delta)c}{\delta} \sum_{i=1}^m q_i \leq \frac{2(1+\delta)c}{\delta} \|R\|$$

which implies the desired result.

**Theorem 3.** *for any  $\delta > 0$ , the algorithm described above is  $(1 + 2\delta)$ -speed,  $O(1)$ -competitive for profit maximization in the unreliable model. In particular  $\|A\|^* \leq (6 + \frac{12}{\delta} + \frac{8}{\delta^2})\|C\|$ .*

*Proof.* The result follows by Lemmas 2 and 5, and setting  $c = 1 + \frac{2}{\delta}$ .

### 3.2 Lower Bounds in the Unreliable Model

It is easily seen that resource augmentation is necessary to obtain  $O(1)$  competitive algorithms in the unreliable model. In fact, by the results of Baruah et al. [1] it follows that every deterministic algorithm is  $\Omega(k)$  competitive even if all the profit functions are of the type  $p_i(t) = p_i$  during  $[0, d_i]$  and 0 thereafter. Here  $k$  is the ratio of the maximum to minimum job density. We can show (proof deferred to the journal version for lack of space) that 1-competitiveness is not possible for any online algorithm, even when it is given faster processors and the job size and profit are bounded.

**Theorem 4.** *Let  $c \geq 1$  be any integer. Consider the profit maximization problem in the unreliable model. Any deterministic  $c$ -speed algorithm is at least  $(1 + \frac{1}{c \cdot 2^c})$ -competitive, even when all jobs are of size 1 and the profit is either 0 or in the range  $[0.5, 1]$ .*

### 3.3 The Reliable Model

We show substantially stronger lower bounds for the profit maximization problem in the reliable model where jobs must be completed by their due dates. Let  $\Delta$  be the ratio of the maximum to minimum job size. Let  $p_i^* = p_i(w_i)$  be the maximum possible profit achievable by a job  $J_i$ , and set  $u_i^* = p_i^*/w_i$ . Let  $k$  denote the maximum to minimum ratio of  $u_i^*$ . The following lower bound states that  $O(1)$ -competitive algorithm is possible only when both  $k$  and  $\Delta$  are constant.

**Theorem 5.** *Any deterministic online algorithm is at least  $\Omega(k \cdot \Delta)$ -competitive in the reliable model of the profit maximization problem.*

The proof of Theorem 5 is deferred to the journal version for lack of space. Next we show that constant competitive ratio is not possible even if we use an arbitrarily large constant speed-up.

**Theorem 6.** *Let  $c \geq 1$  be an integer. Any deterministic  $c$ -speed algorithm is  $\Omega(k^{1/c})$ -competitive in the reliable model of the profit maximization problem.*

*Proof.* For any  $c$ -speed algorithm  $A$ , we release a sequence of at most  $c + 1$  jobs defined as follows. Let  $x \geq 2$  be an integer. A job  $J_1$  is released with  $r_1 = 0$ ,  $w_1 = 1$ , and  $p_1(t) = x$  for  $t \in [0, 1]$  and  $p_1(t) = 0$  otherwise. For  $i = 2, 3, \dots, c+1$ ,  $J_i$  is released if  $A$  sets  $d_{i-1}$  to be at most 1; in that case,  $J_i$  is a job with  $r_i = 0$ ,  $w_i = 1$ , and  $p_i(t) = x^i$  for  $t \in [0, 1]$  and  $p_i(t) = 0$  otherwise. Note that if  $J_{c+1}$  is released, it means that  $A$  sets  $d_i$  to be at most 1 for  $i = 1, \dots, c$ . To meet these due dates,  $A$  must set  $d_{c+1}$  to get greater than 1.

Let  $J_r$  be the last job released. Note that  $A$  sets  $d_r$  to be greater than 1. The total profit of  $A$  is at most  $(x^{r-1} + x^{r-2} + \dots + x) \leq \frac{x^r}{x-1}$ . The adversary can set the due date of  $J_r$  to be 1 and obtain a profit of  $x^r$ . Thus,  $A$  is at least  $(x-1)$ -competitive. Note that the density ratio  $k$  is at most  $x^c$ , that is,  $x \geq k^{1/c}$ .

We also note that a matching  $c$ -speed  $O(k^{1/c})$ -competitive algorithm is achievable using standard techniques. We partition the processor into  $c$  unit speed processors where each part runs jobs of similar density (i.e. within  $k^{1/c}$  of each other). A job is assigned a due date greedily to maximize its profit while maintaining feasibility. The algorithm maintains an online estimate of  $k$ , and merges some classes if  $k$  changes substantially.

## 4 Conclusions

It would be interesting to investigate other due date scheduling problems, from say the surveys [5] and [7], using worst-case analysis to get a better understanding of the effect of the introduction of due dates.

**Acknowledgments.** We would like to thank Steef van de Velde for helpful discussions.

## References

1. Baruah, S., Koren, G., Mishra, B., Raghunathan, A., Rosier, L., Shasha, D.: On-line scheduling in the presence of overload. In: Symposium on Foundations of Computer Science, pp. 100–110. IEEE Computer Society Press, Los Alamitos (1991)
2. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Pruhs, K.: Online weighted flow time and deadline scheduling. *Journal of Discrete Algorithms* 4(3), 339–352 (2006)
3. Fisher, M.: What is the right supply chain for your product. In: *Harvard Business Review*, pp. 105–116 (1997)
4. Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. *JACM* 47, 214–221 (2000)
5. Kaminsky, P., Hochbaum, D.: Due date quotation models and algorithms. In: Leung, J.Y-T. (ed.) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (chapter 20) CRC Press, Inc. (2004)
6. Keskinocak, P., Ravi, R., Tayur, S.: Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues. *Management Science* 47(2), 264–279 (2001)
7. Keskinocak, P., Tayur, S.: Due date management policies. In: Simchi-Levi, D., Wu, S.D., Shen, Z.-J(M.) (eds.) *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*, pp. 485–554. Springer, Heidelberg (2004)
8. Pruhs, K., Sgall, J., Torng, E.: Online scheduling. In: Joseph, Y-T. (ed.) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, CRC Press (2004)
9. Stalk, G.: Time — the next source of competitive advantage. In: *Harvard Business Review*, pp. 41–51 (1988)



# Mechanism Design for Fractional Scheduling on Unrelated Machines

George Christodoulou<sup>1</sup>, Elias Koutsoupias<sup>2</sup>, and Annamária Kovács<sup>1</sup>

<sup>1</sup> Max-Planck-Institut für Informatik, Saarbrücken, Germany  
{gchristo,panni}@mpi-inf.mpg.de

<sup>2</sup> Department of Informatics, University of Athens  
elias@di.uoa.gr

**Abstract.** In this paper, we consider the mechanism design version of the fractional variant of the scheduling problem on unrelated machines. We give a lower bound of  $2 - 1/n$  for any fractional truthful mechanism, while we propose a truthful mechanism that achieves approximation of  $1 + (n - 1)/2$ , for  $n$  machines. We also focus on an interesting family of allocation algorithms, the *task-independent* algorithms. We give a lower bound of  $1 + (n - 1)/2$ , that holds for every (not only monotone) allocation algorithm of this class. Under this consideration, our truthful independent mechanism is the best that we can hope from this family of algorithms.

## 1 Introduction

Mechanism design is an important branch of Microeconomics and in particular of Game Theory. The objective of a mechanism designer is to implement a goal, e.g., to sell an object to a set of potential buyers. The problem derives from the fact that the designer may not be informed about some parameters of the input. These values are controlled by selfish agents that may have incentive to misinform the designer, if this can serve their atomic interests. The mechanism design approach concerns the construction of a game, so that the outcome (equilibrium) of the game is the goal of the designer.

Task scheduling is one of the most important and well-studied problems in Computer Science, as it often arises, in numerous forms, as a subproblem in almost every subfield of Computer Science. One of its most classical and general variants is the scheduling on *unrelated* machines. In this setting, there are  $n$  machines and  $m$  tasks, and the processing time needed by machine  $i$  to perform task  $j$  is determined by the  $t_{ij}$  entry of an  $n \times m$  matrix  $t$ . A common objective is to assign the tasks to the machines in such a way, that the maximum load of the machines (i.e., the makespan) is minimized.

---

<sup>1</sup> In Game-theoretic settings  $n$  is used to denote the number of the players, while in scheduling literature, usually  $m$  is used to denote the cardinality of the machines set. In our case, the aforementioned sets coincide. We prefer to use the former notation, in order to be compatible with the original paper [20] by Nisan and Ronen.

Nisan and Ronen [20] initiated the study of the mechanism design version of scheduling on unrelated machines. In this form of the problem, the processing times that a machine  $i$  needs in order to execute the tasks (vector  $t_i$ ), are *private* values, known only to the corresponding machine. The machines are controlled by selfish agents that aim at satisfying their own interests, and in the particular case they are unwilling to perform any task. In order to motivate them to reveal their actual values, the classical approach adopted by mechanism design, is to introduce side payments, i.e., to hire the machines. A mechanism for this problem consists of an allocation algorithm and a payment scheme. We are interested in bounding the approximation ratio of the mechanism's allocation algorithm.

In the classical version of the problem, each task must be assigned to exactly one machine. The LP-relaxation of the problem, also known as fractional scheduling, concerns the version where instead of being assigned to a single machine, each task can be splitted among the machines. Fractional variations of combinatorial problems have been studied extensively in network optimization, e.g., routing splittable traffic or flow problems.

The fractional scheduling problem can be formulated as a linear program and hence it can be solved in polynomial time. LP-relaxation turns out to be a useful tool in the design of approximation algorithms (both deterministic and randomized)<sup>2</sup>. Furthermore, it turned out to be a powerful technique to provide randomized truthful mechanisms (see e.g. [16,3]). It is natural to ask how powerful LP-relaxation is in the mechanism design framework.

In this paper we consider the mechanism design version of the fractional scheduling on unrelated machines. An interesting fact is that while the offline problem is polynomially solvable, it turns out that in the mechanism design version of the problem it cannot be approximated within a constant factor, even by non-polynomial mechanisms (see Sec. 3). This means, that the additional properties that the allocation of a mechanism needs to satisfy in contrast to a simple algorithm (cf. Sec. 2), do not allow us to achieve an exact solution, even in non-polynomial time. Lower bounding fractional mechanisms is a nice approach to lower bound randomized (and deterministic) mechanisms of the integral case, as splitting a job is clearly a more radical solution than randomly assigning it.

We are particularly interested in a family of mechanisms that we call *task-independent*. A task-independent algorithm is any algorithm that in order to allocate task  $j$ , only considers the processing times  $t_{ij}$ , that concern the particular task. Such a consideration is motivated by the fact that (to the best of our knowledge) all the known positive results for this problem (e.g., see the mechanisms in [18,20]), and in addition the mechanism that we propose in this paper, belong to this family of mechanisms. The question that we address here is: *how far can we go with task-independent algorithms?*

## 1.1 Related Work

Scheduling on unrelated machines is a classical NP-hard problem. Lenstra et al. [17] gave a 2-approximation polynomial time algorithm, while they also proved

<sup>2</sup> In fact, it has been used in order to obtain the 2-approximation algorithm in [17].

that the problem cannot be approximated (in polynomial time) within a factor less than  $3/2$ . The mechanism design version of the problem originates in the seminal work of Nisan and Ronen [20]. They gave a  $n$ -approximation truthful mechanism and a lower bound of 2, while they conjectured the actual bound to be  $n$ . Christodoulou et al. [9] improved the lower bound to  $1 + \sqrt{2}$ . Narrowing the gap between the lower and the upper bound still remains a big open question.

Randomization usually reduces the approximation ratio and that is also the case for this problem. Nisan and Ronen [20] proposed a randomized mechanism for 2 machines with approximation ratio  $7/4$ . Recently, Mu'alem and Schapira [18] generalized this mechanism for  $n$  machines and achieved a  $7n/8$  randomized truthful mechanism. In the same work, they also gave a lower bound of  $2 - 1/n$  for randomized mechanisms. Notice that all the known lower bounds for this problem (both deterministic and randomized) follow due to the infrastructure of truthful mechanisms, and do not reside in any computational assumption; consequently they hold even for non polynomial time mechanisms.

Scheduling on related machines, from the mechanism design point of view, was first studied by Archer and Tardos [4]. In this variant of the problem, the private parameter for each machine, is a single value (its speed). Archer and Tardos [4] characterized the class of truthful mechanisms for this setting, in terms of a monotonicity condition of the mechanism's allocation algorithm. A similar characterization for one-parameter mechanism design problems (single item auction) can also be found in Myerson [19]. For this problem, it turns out that the optimal allocation algorithm can be modified to be a truthful mechanism. Archer and Tardos [4] gave a randomized truthful 3-approximation algorithm, which was later improved to a 2-approximation by Archer [2]. Andelman et al. [1] gave the first deterministic polynomial mechanism for the problem, with an approximation ratio of 5. Kovács [13] improved this by giving a 3-approximation deterministic truthful mechanism, while finally the ratio was reduced to 2.8 [14].

In the field of Combinatorial Auctions, a wide variety of combinatorial optimization problems has been considered from the mechanism design point of view (see for example [3,6,8,10,5,11] and references within). In this context, Saks and Yu [21] characterized the class of truthful mechanisms for combinatorial auctions with convex valuations, generalizing results of [7,12,15].

## 1.2 Our Results

In this paper, we consider the mechanism design version of fractional scheduling on unrelated machines. We give a  $2 - 1/n$  lower bound on the approximation ratio, that can be achieved by any truthful mechanism. This result shows that even in the case of such a problem, for which the offline version can be exactly solved in polynomial time, its mechanism design analog may turn out to be impossible to approximate, even by non-polynomial mechanisms. Notice that giving a lower bound for fractional mechanisms is another way to obtain lower bounds for randomized mechanisms for the integral case. Consequently, our  $2 - 1/n$  lower bound extends the lower bounds in [18] to the class of fractional mechanisms. Note that a fractional mechanism is more powerful than a randomized

mechanism for the integral case, since it has the flexibility to split a task into many machines, while a randomized mechanism, finally, has to assign the whole task to a machine, and this affects its approximation ratio.

In the positive direction, we give a truthful mechanism with approximation ratio  $3/2$  for 2 machines, which matches our lower bound. This is the first new tight bound that we have for any variant of the problem, after the tight bound of 2 in the integral case, obtained for 2 machines in [20]. The generalization of our mechanism for  $n$  machines gives us an approximation ratio of  $1 + (n - 1)/2$ .

Next we turn our attention to a family of mechanisms that we call *task-independent*. This family consists of mechanisms, where the decision for the assignment of a task, depends only on the processing times that concern the particular task (*time column w.r.t. the task*). Considering task-independence is motivated by the fact that all known 'reasonable' deterministic and randomized mechanisms for this problem are task-independent. Furthermore, this sort of independence has attractive properties: easy to design by applying methods for one-parameter auctions, fits well with on-line settings, where tasks may appear one-by-one. It is natural to ask if there is room for improvement on the approximation ratio by use of such mechanisms. We extend this question for the class of task-independent *algorithms* that need not satisfy the additional properties imposed by truthfulness. We give a lower bound of  $1 + (n - 1)/2$  on the approximation ratio of any algorithm that belongs to this class. Our mechanism is also task-independent, and hence is optimal over this family of algorithms.

## 2 Problem Definition

In this section, we fix the notation that we will use throughout this paper, furthermore we give some preliminary definitions and cite relevant results.

There are  $n$  machines and  $m$  tasks. Each machine  $i \in [n]$  needs  $t_{ij}$  units of time to perform task  $j \in [m]$ . We denote by  $t_i$  the row vector corresponding to machine  $i$ , and by  $t^j$  the column vector of the running times of task  $j$ . We assume that each machine  $i \in [n]$  is controlled by a selfish agent that is unwilling to perform any operation, and vector  $t_i$  is private information known only to her. The vector  $t_i$  is also called the *type* of agent  $i$ .

Any mechanism defines for each player  $i$  a set  $A_i$  of available strategies, the player (agent) can choose from. We will consider *direct revelation* mechanisms, i.e.,  $A_i = T_i$  for all  $i$ , meaning that the players strategies are to simply report their types to the mechanism. In general,  $T_i$  consists of all possible vectors  $b_i \in \mathbb{R}_+^m$ , that is, a player may report a false vector  $b_i \neq t_i$ , if this serves his interests.

A mechanism  $M = (x, p)$  consists of two parts:

**An allocation algorithm:** The allocation algorithm  $x$ , depends on the players' bids  $b = (b_1, \dots, b_n)$ , with  $0 \leq x_{ij} \leq 1$  denoting the fraction of task  $j$  that is assigned to the machine  $i$ . In the unsplittable case, these variables take only integral values  $x_{ij} \in \{0, 1\}$ . Every task must be completely assigned to the machines' set, so  $\sum_{i \in [n]} x_{ij} = 1, \quad \forall j \in [m]$ .

**A payment scheme:** The payment scheme  $p = (p_1, \dots, p_n)$ , also depends on the bid values  $b$ . The functions  $p_1, \dots, p_n$  stand for the payments that the mechanism hands to each agent.

The *utility*  $u_i$  of a player  $i$  is the payment that he gets minus the *actual* time that he needs in order to execute the set of tasks assigned to her,  $u_i(b) = p_i(b) - \sum_{j \in [m]} t_{ij} x_{ij}(b)$ . We are interested in *truthful* mechanisms. A mechanism is truthful, if for every player, reporting his true type is a *dominant strategy*. Formally,

$$u_i(t_i, b_{-i}) \geq u_i(t'_i, b_{-i}), \quad \forall i \in [n], t'_i \in T_i, b_{-i} \in T_{-i},$$

where  $T_{-i}$  denotes the possible types of all players disregarding  $i$ .

We remark here, that once we adopt the solution concept of dominant strategies, focusing on direct revelation and in particular on truthful mechanisms is not at all restrictive, due to the *Revelation Principle*. Roughly, the Revelation Principle states that any problem that can be implemented by a mechanism with dominant strategies, can also be implemented by a truthful mechanism (cf. [20]).

The objective function that we consider, in order to evaluate the performance of a mechanism's allocation algorithm  $x$ , is the maximum load of a machine (makespan), with respect to the real time matrix  $t$ . When we refer to the makespan of a mechanism, we mean the makespan of its allocation algorithm with respect to the input  $t$ , and we denote it by  $Mech(t) = \max_{i \in [n]} \sum_{j \in [m]} t_{ij} x_{ij}$ . Since we aim at minimizing the makespan, the optimum is  $Opt(t) = \min_x \max_{i \in [n]} \sum_{j \in [m]} t_{ij} x_{ij}$ . We are interested in the approximation ratio of the mechanism's allocation algorithm. A mechanism  $M$  is  $c$ -approximate, if  $c \geq Mech(t)/Opt(t) \quad \forall t \in T$ .

Although our mechanism is polynomially computable, we do not aim at minimizing the running time of the algorithm; we are looking for mechanisms with low approximation ratio. Our lower bounds also don't make use of any computational assumptions.

A useful characterization of truthful mechanisms in terms of the following monotonicity condition, helps us to get rid of the payments and focus on the properties of the allocation algorithm.

**Definition 1.** An allocation algorithm is called *monotone*<sup>3</sup> if it satisfies the following property: for every two sets of tasks  $t$  and  $t'$  which differ only on machine  $i$  (i.e., on the  $i$ -th row) the associated allocations  $x$  and  $x'$  satisfy  $(x_i - x'_i) \cdot (t_i - t'_i) \leq 0$ , where  $\cdot$  denotes the dot product of the vectors, that is,  $\sum_{j \in [m]} (x_{ij} - x'_{ij})(t_{ij} - t'_{ij}) \leq 0$ .

The following theorem states that every truthful mechanism has to satisfy the monotonicity condition. It was used by Nisan and Ronen [20] in order to obtain their lower bounds.

**Theorem 1.** Every truthful mechanism is monotone.

<sup>3</sup> Also known as *weakly monotone*.

Saks and Yu [21] proved that monotonicity is also a sufficient condition, for the combinatorial auctions setting with convex valuations (*i.e. there exist payments that can make a monotone algorithm into a truthful mechanism*).

For the one-parameter case, *i.e.*, when every agent has a single value to declare (*e.g.*, the speed of her machine), Myerson [19] (*for auction setting*) and Archer and Tardos [4] (*for scheduling setting*), showed that the monotonicity of the (allocation) algorithm is a necessary and sufficient condition for the existence of a truthful payment scheme. In this case they also provide an explicit formula for the payments. In their theorem cited below, the notion of a *decreasing output function*, corresponds to a monotone algorithm in the one-parameter setting.

**Theorem 2.** [19,4] *The output function admits a truthful payment scheme if and only if it is decreasing. In this case the mechanism is truthful if and only if the payments  $p_i(b_i, b_{-i})$  are of the form*

$$h_i(b_{-i}) + b_i x_i(b_i, b_{-i}) - \int_0^{b_i} x_i(u, b_{-i}) du$$

where the  $h_i$  are arbitrary functions.

### 3 Lower Bound for Truthful Mechanisms

Here we will give a lower bound on the approximation ratio of any fractional truthful mechanism.

**Theorem 3.** *There is no deterministic truthful mechanism that can achieve an approximation ratio better than  $2 - \frac{1}{n}$ , where  $n$  is the number of the machines.*

*Proof.* Let  $t$  be the actual time matrix of the players as below

$$t_{ij} = \begin{cases} 0, & j = i \\ 1, & j = n + 1 \\ A, & \text{otherwise} \end{cases}$$

and  $x = x(t)$  be the corresponding allocation that a truthful mechanism  $M = (x, p)$  gives with respect to  $t$ . For significantly large values of  $A$ , player  $i$  gets substantially the whole portion of task  $i$ , otherwise the approximation ratio is high, *e.g.*, for  $A = \frac{2}{\delta}$ , every player  $i$  should get a portion greater than  $1 - (n-1)\delta$ , otherwise the approximation ratio is at least 2.

Clearly, there is a player  $k \in [n]$ , with  $x_{kn+1} \geq \frac{1}{n}$ . Now let's consider how the allocation algorithm of the mechanism behaves if the following time matrix is given as input

$$t'_{ij} = \begin{cases} \frac{1}{n-1}, & i = k, j = i \\ 1 - \epsilon, & i = k, j = n + 1 \\ t_{ij}, & \text{otherwise} \end{cases}$$

The following claim states that due to monotonicity, the mechanism cannot assign to player  $k$  a substantially smaller portion of the  $n + 1^{\text{st}}$  task than  $\frac{1}{n}$ .

*Claim.* If  $x_{kn+1} \geq \frac{1}{n}$ , then for the allocation  $x' = x(t')$  on input  $t'$  it holds that  $x'_{kn+1} \geq \frac{1}{n} - \epsilon$ .

*Proof.* Due to Theorem [1](#) we have that for every player  $i \in [n]$ , it holds that

$$\sum_{j \in [m]} (t_{ij} - t'_{ij})(x_{ij} - x'_{ij}) \leq 0$$

and by applying this to the  $k$ -th player we get

$$\left(0 - \frac{1}{n-1}\right) (x_{kk} - x'_{kk}) + (1 - 1 + \epsilon)(x_{kn+1} - x'_{kn+1}) \leq 0,$$

from which we get

$$x'_{kn+1} \geq x_{kn+1} + \frac{x'_{kk} - x_{kk}}{\epsilon(n-1)} \geq x_{kn+1} - \frac{\delta}{\epsilon} \geq \frac{1}{n} - \frac{\delta}{\epsilon}$$

and for  $\delta = \epsilon^2$  we finally obtain

$$x'_{kn+1} \geq \frac{1}{n} - \epsilon$$

On the other hand, an optimal allocation  $x^*$  for  $t'$  is □

$$x^*_{ij} = \begin{cases} 1, & j = i \\ 0, & i = k, j = n + 1 \\ \frac{1}{n-1}, & i \neq k, j = n + 1 \\ 0, & \text{otherwise} \end{cases}$$

with makespan  $1/(n-1)$ , while the mechanism gives player  $k$  a total load of at least

$$(1 - (n-1)\delta) \frac{1}{n-1} + \left(\frac{1}{n} - \epsilon\right) (1 - \epsilon) > \frac{1}{n-1} + \frac{1}{n} - \delta - \epsilon \left(\frac{n+1}{n}\right).$$

For arbitrary small  $\epsilon$ , this finally gives an approximation ratio of at least  $2 - \frac{1}{n}$ . □

## 4 The Truthful Mechanism

We describe a truthful mechanism, called SQUARE, for the fractional scheduling problem, with approximation ratio  $1 + \frac{n-1}{2}$ . On two machines this ratio becomes  $3/2$ , so in this case SQUARE has the best possible worst case ratio w.r.t. truthful mechanisms. Furthermore, in Section [5](#) we will show that for arbitrary number of machines, our mechanism is optimal among the so called *task-independent* algorithms.

Next, we define the mechanism SQUARE =  $(x^{Sq}, p^{Sq})$ [4](#). Recall that  $b_{ij}$  is the reported value for  $t_{ij}$ , the actual execution time of task  $j$  on machine  $i$ .

<sup>4</sup> In most of the section we will omit the superscripts  $Sq$ .

**Definition 2 (The mechanism SQUARE=  $(x^{Sq}, p^{Sq})$ )**

**Allocation algorithm:** Let  $b^j = (b_{1j}, b_{2j}, \dots, b_{nj})^T$  be the  $j$ th column-vector of the input matrix. If  $b^j$  has at least one zero coordinate, then SQUARE distributes the  $j$ th task among machines having zero execution time arbitrarily. If  $b_{ij} \neq 0$  ( $i \in [n]$ ), then the fraction of the  $j$ th task allocated to machine  $i$  is

$$x_{ij}^{Sq}(b) = x_{ij}(b) = \frac{\prod_{k \neq i} b_{kj}^2}{\sum_{l=1}^n \prod_{k \neq l} b_{kj}^2}. \quad (1)$$

**Payment scheme:** Let the constants  $c_{ij}$  be defined as

$$c_{ij} = \frac{\prod_{k \neq i} b_{kj}}{\sqrt{\sum_{l \neq i} \prod_{k \neq l, i} b_{kj}^2}},$$

then the payments  $p^{Sq} = (p_1, \dots, p_n)$  to the agents are

$$p_i(b) = \sum_{j=1}^m \left( b_{ij} \cdot \frac{c_{ij}^2}{b_{ij}^2 + c_{ij}^2} + c_{ij} \cdot \frac{\pi}{2} - c_{ij} \arctan \frac{b_{ij}}{c_{ij}} \right).$$

The algorithm  $x^{Sq}$  of SQUARE allocates the tasks individually (independently), and so that the assigned fractions of a task are inversely proportional to the squares of (declared) execution times w.r.t. each machine. For instance, for two machines  $\square$  boils down to

$$x_{1j} = \frac{b_{2j}^2}{b_{1j}^2 + b_{2j}^2}; \quad x_{2j} = \frac{b_{1j}^2}{b_{1j}^2 + b_{2j}^2}.$$

For arbitrary  $n$  it is obvious that  $0 \leq x_{ij} \leq 1$ , and  $\sum_{i=1}^n x_{ij} = 1$ . It is easy to see that SQUARE is monotone: Let the input matrix  $b$  be changed only on the  $i$ th row, that is, for any fixed task  $j$ , just the entry  $b_{ij}$  may change. Assume first that in the column-vector  $b^j$  all execution times are nonzero. Observe that the variable  $b_{ij}$  appears only in the denominator of the expression  $\square$ , namely as  $b_{ij}^2$ , having a positive coefficient. Thus,  $x_{ij}$  does not increase when  $b_{ij}$  increases, and vice versa. It is easy to see that the same holds if in  $b^j$  there are zero entries other than  $b_{ij}$ , and similarly, if  $b_{ij}$  was, or just became the only zero entry. Thus, we obtained that for every single one-parameter problem  $b^j$ , the assignment is monotone, and this, in turn, implies weak monotonicity (see Definition  $\square$ ) for  $x^{Sq}$ .

Now consider  $p^{Sq}$ . For two machines, the constant  $c_{ij}$  is simply the bid of the other machine for this job, that is,  $c_{1j} = b_{2j}$  and  $c_{2j} = b_{1j}$ . For more machines,  $c_{ij}$  would be the 'bid' of a single other machine, if we replaced the machines  $[n] \setminus \{i\}$  with one machine. The payment  $p_i(b)$  is simply defined to be the sum of the payments that agent  $i$  would get for performing each (fractional) task independently, as determined for truthful mechanisms for one-parameter agents by Theorem  $\square$ :

$$p_i(b_i, b_{-i}) = h_i(b_{-i}) + b_i x_i(b_i, b_{-i}) - \int_0^{b_i} x_i(u, b_{-i}) du.$$



Here the  $h_i(b_{-i})$  are arbitrary constants. If we want that the so called *voluntary participation* [4] of the players is ensured (i.e., it is worth taking part in the game), then  $h_i$  can be chosen to be  $h_i = \int_0^\infty x_i(u, b_{-i}) du$ , so that finally we get

$$p_i(b_i, b_{-i}) = b_i x_i(b_i, b_{-i}) + \int_{b_i}^\infty x_i(u, b_{-i}) du,$$

for the one-parameter case. Applying this formula for each task, we obtain the above definition of the payments.

**Theorem 4.** *The mechanism SQUARE is truthful.*

*Proof.* To put it short, the theorem follows from the fact that SQUARE is the sum of  $m$  truthful mechanisms for the one-parameter problem (note that the total execution time (load), the total payment, thus also the total utility is the sum of the respective amounts for the single tasks). For each  $j$ , the mechanism  $(x^j, p^j)$  is truthful, since  $x^j$  is a monotone algorithm, and we defined  $p^j = (p_{1j}, \dots, p_{nj})^T$  according to Theorem [2]. We do not include an elementary proof here.  $\square$

**Approximation Ratio.** Let  $Squ(t)$  be the makespan of the schedule produced by SQUARE on input  $t$ , and  $Opt(t)$  denote the optimum makespan. In what follows, we show that  $Squ(t)/Opt(t) \leq 1 + \frac{n-1}{2}$  for any matrix  $t$ . The next lemma will largely simplify the upper-bound proof. The proof of the lemma is omitted.

**Lemma 1.** *If there exists an input instance  $t$ , such that  $Squ(t)/Opt(t) = \alpha$ , then there also exists an instance  $t^*$ , for which  $Squ(t^*)/Opt(t^*) = \alpha$ , moreover there is an optimal allocation of  $t^*$  that does not split any job.*

**Theorem 5.** *For the approximation ratio of SQUARE,  $\frac{Squ(t)}{Opt(t)} \leq 1 + \frac{n-1}{2}$  holds, where  $n$  denotes the number of machines, and  $t$  is an arbitrary set of input tasks.*

*Proof.* Consider the input  $t$ . Due to Lemma [1], we can assume that the (indices of) tasks are partitioned into the sets  $J_1, J_2, \dots, J_n$ , so that there is an optimal allocation OPT where job  $t^j$  is allocated completely to machine  $i$ , if and only if  $j \in J_i$ . We can also assume that  $t_{ij} > 0$  for all  $i$  and  $j$ . Otherwise we would have a job that adds zero execution time to the makespan in both the allocation of SQUARE, and of OPT, and removing this job from the input would not affect the approximation ratio. For the optimum makespan it holds that

$$Opt(t) = \max_{i \in [n]} \sum_{j \in J_i} t_{ij}. \quad (2)$$

For the running time of an arbitrary machine  $i$  in SQUARE, we have

$$Squ_i(t) = \sum_{r=1}^n \sum_{j \in J_r} x_{ij}(t) t_{ij},$$

where the  $x_{ij}(t)$  are defined by (11). We decompose the above expression as follows:

$$Squ_i(t) = \sum_{j \in J_i} x_{ij} t_{ij} + \sum_{r \neq i} \sum_{j \in J_r} x_{ij} t_{ij}.$$

We can upper bound the first sum using (2), and the fact that  $x_{ij} \leq 1$ :

$$\sum_{j \in J_i} x_{ij} t_{ij} \leq \sum_{j \in J_i} 1 \cdot t_{ij} \leq Opt(t).$$

Next we upper bound every sum of the form  $\sum_{j \in J_r} x_{ij} t_{ij}$  ( $r \neq i$ ), by  $\frac{1}{2} \cdot Opt(t)$ . Since there are  $n - 1$  such sums, this will prove that

$$Squ_i(t) \leq Opt(t) + (n - 1) \cdot \frac{1}{2} \cdot Opt(t) = \left(1 + \frac{n - 1}{2}\right) \cdot Opt(t).$$

Since  $i$  was an arbitrary machine, eventually this implies

$$Squ(t) = \max_{i \in [n]} Squ_i(t) \leq \left(1 + \frac{n - 1}{2}\right) \cdot Opt(t).$$

The bound  $\sum_{j \in J_r} x_{ij} t_{ij} \leq \frac{1}{2} \cdot Opt(t)$  can be shown as follows:

$$\begin{aligned} \sum_{j \in J_r} x_{ij} t_{ij} &= \sum_{j \in J_r} \frac{\prod_{k \neq i} t_{kj}^2}{\sum_{l=1}^n \prod_{k \neq l} t_{kj}^2} \cdot t_{ij} = \sum_{j \in J_r} \frac{t_{ij} t_{rj} \prod_{k \neq i, r} t_{kj}^2}{\sum_{l=1}^n \prod_{k \neq l} t_{kj}^2} \cdot t_{rj} = \\ &= \sum_{j \in J_r} \frac{t_{ij} t_{rj}}{t_{ij}^2 + t_{rj}^2 + \sum_{l \neq i, r} t_{ij}^2 t_{rj}^2 / t_{ij}^2} t_{rj} \leq \sum_{j \in J_r} \frac{t_{ij} t_{rj}}{t_{ij}^2 + t_{rj}^2} t_{rj} \leq \sum_{j \in J_r} \frac{1}{2} t_{rj} \leq \frac{1}{2} Opt(t), \end{aligned}$$

where the inequality  $\frac{t_{ij} t_{rj}}{t_{ij}^2 + t_{rj}^2} \leq \frac{1}{2}$  holds for any two nonzero real numbers; the last inequality is implied by (2).  $\square$

**Corollary 1.** *For two machines the truthful mechanism SQUARE has approximation ratio  $3/2$ , which is the best worst case ratio we can expect from any truthful mechanism for the fractional scheduling problem.*

## 5 Lower Bound for Independent Algorithms

In this section we prove a lower bound of  $1 + \frac{n-1}{2}$  for the worst case ratio of independent fractional algorithms. An algorithm is independent, if it allocates the tasks independently of each-other, or formally:

**Definition 3.** *An allocation algorithm  $x$  is called task-independent, or simply independent, if the following holds: If  $t$  and  $t'$  are two  $n \times m$  input matrices, such that for the  $j$ th task  $t_{ij} = t'_{ij}$  ( $\forall i \in [n]$ ), then for this task it also holds that  $x_{ij} = x'_{ij}$  ( $\forall i \in [n]$ ).*

It is remarkable, that the currently known best mechanisms (in fact, any 'reasonable' mechanism we know of) are all independent, in the integral, the randomized, and the fractional case. It is not difficult to come up with independent (suboptimal) algorithms, which are also weakly monotone. However it seems to be an intriguing question, whether there exist non-independent, and still monotone algorithms having better approximation ratio than the best independent ones. We note that in the integral case it is easy to construct an instance with  $n$  machines and  $n^2$  tasks, that proves a lower bound of  $n$  (i.e., tight bound) for independent algorithms.

**Theorem 6.** *If  $x$  is an independent fractional allocation algorithm for the unrelated machines problem, then it has approximation ratio of at least  $1 + \frac{n-1}{2}$ , where  $n$  denotes the number of machines.*

*Proof.* In order to obtain the lower bound, consider the following input matrix with  $n \geq 2$  machines and  $m = 1 + \binom{n}{2}$  tasks. The first task is numbered by  $j = 0$ ; furthermore, for all  $\binom{n}{2}$  possible pairs of machines  $g < h$  there is a task  $j_{gh}$  :

$$t_{ij} = \begin{cases} 0, & j = 0 \\ 1, & j = j_{gh}, i \in \{g, h\} \\ A, & \text{otherwise} \end{cases}$$

Obviously, by setting  $A$  large enough, we can make it sure – like in the proof of Theorem 3 – that the corresponding share of a player of a certain task is arbitrarily small, otherwise the approximation ratio gets too large. That is, we can assume that the bulk of any job is allocated to the machines having execution time 1 for this job.

Let us consider an arbitrary independent algorithm  $x$ . Observe that no matter how  $x$  allocates the above tasks, the total running time of all the jobs cannot be less than  $\binom{n}{2}$ . Thus, there exists a machine, say machine  $k$ , with running time at least  $\binom{n}{2}/n = \frac{n-1}{2}$ . Now we modify the instance  $t$  to  $t'$  : we keep the original execution times of tasks that had running time 1 on machine  $k$ , and zero out all other  $t_{ij}$ ; furthermore, task 0 will now have execution time 1 on machine  $k$ , and  $A$  on other machines.

$$t'_{ij} = \begin{cases} 1, & j = 0, i = k \\ A, & j = 0, i \neq k, \\ t_{ij}, & j = j_{gh} \text{ and } g = k \text{ or } h = k \\ 0, & \text{otherwise} \end{cases}$$

As noted above, on instance  $t$  at least  $\frac{n-1}{2} - \epsilon$  running time on machine  $k$  was due to jobs that have execution time 1 on this machine. Since  $x$  is independent, on instance  $t'$  the machine gets the same allocation over these jobs, and also gets a  $(1 - \epsilon)$  fraction of job 0, achieving a running time of at least  $1 + (n-1)/2 - 2\epsilon$ , for any  $\epsilon > 0$ . On the other hand, it is clear that the optimal allocation has makespan 1.  $\square$

**Corollary 2.** *The mechanism SQUARE has optimal approximation ratio among all independent mechanisms.*

One can show that among all allocations where the distribution of task  $j$  is inversely proportional to  $(t_{1j}^\alpha, t_{2j}^\alpha, \dots, t_{nj}^\alpha)$  for some  $\alpha > 0$ , the above optimal approximation ratio is obtained if and only if  $\alpha = 2$ .

## References

1. Andelman, N., Azar, Y., Sorani, M.: Truthful approximation mechanisms for scheduling selfish related machines. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 69–82. Springer, Heidelberg (2005)
2. Archer, A.: Mechanisms for Discrete Optimization with Rational Agents. PhD thesis, Cornell University (January 2004)
3. Archer, A., Papadimitriou, C.H., Talwar, K., Tardos, É.: An approximate truthful mechanism for combinatorial auctions with single parameter agents. In: SODA, pp. 205–214 (2003)
4. Archer, A., Tardos, É.: Truthful mechanisms for one-parameter agents. In: FOCS, pp. 482–491 (2001)
5. Babaioff, M., Lavi, R., Pavlov, E.: Mechanism design for single-value domains. In: AAAI, pp. 241–247 (2005)
6. Bartal, Y., Gonen, R., Nisan, N.: Incentive compatible multi unit combinatorial auctions. In: TARK, pp. 72–87 (2003)
7. Bikhchandani, S., Chatterji, S., Lavi, R., Mu’alem, A., Nisan, N., Sen, A.: Weak monotonicity characterizes deterministic dominant strategy implementation. *Econometrica* 74(4), 1109–1132 (2006)
8. Briest, P., Krysta, P., Vöcking, B.: Approximation techniques for utilitarian mechanism design. In: STOC, pp. 39–48 (2005)
9. Christodoulou, G., Koutsoupias, E., Vidali, A.: A lower bound for scheduling mechanisms. In: SODA, pp. 1163–1170 (2007)
10. Dobzinski, S., Nisan, N., Schapira, M.: Approximation algorithms for combinatorial auctions with complement-free bidders. In: STOC, pp. 610–618 (2005)
11. Dobzinski, S., Nisan, N., Schapira, M.: Truthful randomized mechanisms for combinatorial auctions. In: STOC, pp. 644–652 (2006)
12. Gui, H., Müller, R., Vohra, R.V.: Dominant strategy mechanisms with multidimensional types. In: Computing and Markets (2005)
13. Kovács, A.: Fast monotone 3-approximation algorithm for scheduling related machines. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 616–627. Springer, Heidelberg (2005)
14. Kovács, A.: Fast Algorithms for Two Scheduling Problems. PhD thesis, Universität des Saarlandes (2007)
15. Lavi, R., Mu’alem, A., Nisan, N.: Towards a characterization of truthful combinatorial auctions. In: FOCS, pp. 574–583 (2003)
16. Lavi, R., Swamy, C.: Truthful and near-optimal mechanism design via linear programming. In: FOCS, pp. 595–604 (2005)
17. Lenstra, J.K., Shmoys, D.B., Tardos, É.: Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming* 46(1), 259–271 (1990)
18. Mu’alem, A., Schapira, M.: Setting lower bounds on truthfulness. In: SODA, pp. 1143–1152 (2007)

19. Myerson, R.B.: Optimal auction design. *Mathematics of Operations Research* 6(1), 58–73 (1981)
20. Nisan, N., Ronen, A.: Algorithmic mechanism design. *Games and Economic Behavior* 35, 166–196 (2001)
21. Saks, M.E., Yu, L.: Weak monotonicity suffices for truthfulness on convex domains. In: *EC*, pp. 286–293 (2005)

# Estimating Sum by Weighted Sampling

Rajeev Motwani<sup>1</sup>, Rina Panigrahy<sup>2</sup>, and Ying Xu<sup>1,\*</sup>

<sup>1</sup> Dept of Computer Science, Stanford University, USA

<sup>2</sup> Microsoft Research, Mountain View, CA, USA

{rajeev,xuying}@cs.stanford.edu, rina@microsoft.com

**Abstract.** We study the classic problem of estimating the sum of  $n$  variables. The traditional uniform sampling approach requires a linear number of samples to provide any non-trivial guarantees on the estimated sum. In this paper we consider various sampling methods besides uniform sampling, in particular sampling a variable with probability proportional to its value, referred to as *linear weighted sampling*. If only linear weighted sampling is allowed, we show an algorithm for estimating sum with  $\tilde{O}(\sqrt{n})$  samples, and it is almost optimal in the sense that  $\Omega(\sqrt{n})$  samples are necessary for any reasonable sum estimator. If both uniform sampling and linear weighted sampling are allowed, we show a sum estimator with  $\tilde{O}(\sqrt[3]{n})$  samples. More generally, we may allow general weighted sampling where the probability of sampling a variable is proportional to any function of its value. We prove a lower bound of  $\Omega(\sqrt[3]{n})$  samples for any reasonable sum estimator using general weighted sampling, which implies that our algorithm combining uniform and linear weighted sampling is an almost optimal sum estimator.

## 1 Introduction

We consider the classic problem of estimating the sum (or equivalently, the average) of  $n$  non-negative variables. This problem has numerous important applications in various areas of computer science, statistics and engineering. Measuring the exact value of each variable incurs some cost, so people want to get a reasonable estimator of the sum while measure as few variables as possible.

In the traditional setting, only uniform sampling is used, i.e. each time we can sample one variable uniformly at random and ask its value. Under this setting it is easy to see that any reasonable estimator requires a linear sample size if the underlying distribution is arbitrary. Consider the following two instances of inputs: in the first input all variables are 0, while in the second input all are 0 except one variable  $x_1$  is a large number. Any sampling scheme cannot distinguish the two inputs until it sees  $x_1$ , and with uniform sampling it takes linear samples

---

\* Rajeev Motwani is supported in part by NSF Grants EIA-0137761 and ITR-0331640, and grants from Media-X and SNRC. Rina Panigrahy's work was done when he was at Stanford, and he was supported by Stanford Graduate Fellowship. Ying Xu is supported in part by a Stanford Graduate Fellowship and NSF Grants EIA-0137761 and ITR-0331640.

to hit  $x_1$ . We defer the formal definition of “reasonable estimator” to Section 2, but intuitively we cannot get a good estimator if we cannot distinguish the two inputs.

In this paper, we study the problem of estimating sum using other sampling methods besides uniform sampling. For example, suppose we now allow sampling a variable with probability proportional to its value, which we refer to as *linear weighted sampling*; in Section 1.1 we will discuss applications where such sampling is feasible. Using linear weighted sampling one sample is sufficient to distinguish the above two inputs, and it seems plausible that generally we can get good sum estimators with less samples using such sampling method. In this paper we show an algorithm for sum estimation with  $\tilde{O}(\sqrt{n})$  samples using only linear weighted sampling, and it is almost optimal in the sense that  $\Omega(\sqrt{n})$  samples are necessary for any reasonable estimator using only linear weighted sampling. Our algorithm assumes no prior knowledge about the input distribution.

Next, if we use both uniform sampling and linear weighted sampling, we can further reduce the number of samples needed. We present a sum estimator with  $\tilde{O}(\sqrt[3]{n})$  samples using a combination of the two sampling methods, and prove a lower bound of  $\Omega(\sqrt[3]{n})$  samples.

More generally, we may allow sampling where the probability of sampling a variable can be proportional to any function of its value (the function does not depend on  $n$ ), referred as to *(general) weighted sampling*. While we are not sure whether general sampling is feasible in real applications, we show a negative result that such extra power does not provide a better estimator: we prove a lower bound of  $\Omega(\sqrt[3]{n})$  samples for any reasonable sum estimator, using any combination of general weighted sampling methods. This implies that combining uniform and linear weighted sampling gives an almost optimal sum estimator (up to a poly-log factor), hence there is no need to pursue fancier sampling methods in this family for the purpose of estimating sum.

## 1.1 Applications

The problem of estimating sum is a classic problem with wide applications in various areas, and linear weighted sampling is a natural sampling method feasible in many applications. In particular, if we want to estimate the total number of some objects in a system and those objects fall into disjoint classes, then the problem becomes estimating the sum of variables with each variable indicating the number of objects in one class; if uniform sampling of the objects is possible, then linear weighted sampling can be implemented by sampling an object uniformly at random and returning the class of the sampled object.

One such application is estimating search engine index sizes or the web size, which has aroused interests in both academic and industrial world in recent years (see for example [12][13][10][6][4]). One method used in those papers is to partition the search index (web) into domains (web servers), and estimate the sum of those domain (server) sizes. It is relatively easy to get the total domain (web server) number  $n$  (either by uniformly sampling IP space or people publish this number periodically). For example in 1999 Lawrence and Giles estimated the

number of web servers to be 2.8 million by randomly testing IP addresses; then they exhaustively crawled 2500 web servers and found that the mean number of pages per server was 289, leading to an estimate of the web size of 800 million [13]. Lawrence and Giles essentially used uniform sampling to estimate the sum, however, the domain size distribution is known to be highly skewed and uniform sampling has high variance for such inputs. We can also do linear weighted sampling: uniformly sample a page from the web or a search engine index (the technique of uniform sampling a page from the web/index has been studied in for example [11,3]) and take the domain of the page, then the probability of sampling a domain is proportional to its size. Then we can apply the techniques in this paper, which shall provide a more accurate estimate than using only uniform sampling.

## 1.2 Related Work

Estimating the sum of  $n$  variables is a classical statistical problem. For the case where all the variables are between  $[0, 1]$ , an additive approximation of the mean can be easily computed by taking a random sample of size  $O(\frac{1}{\epsilon^2} \lg \frac{1}{\delta})$  and computing the mean of samples; [7] proves a tight lower bound on the sample size. However, uniform sampling works poorly on heavily tailed inputs when the variables are from a large range, and little is known beyond uniform sampling.

Weighted sampling is also known as “importance sampling”. General methods of estimating a quantity using importance sampling have been studied in statistics (see for example [14]), but the methods are either not applicable here or less optimal. To estimate a quantity  $h_\pi = \sum \pi(i)h(i)$ , importance sampling generates independent samples  $i_1, i_2, \dots, i_N$  from a distribution  $p$ . One estimator for  $h_\pi$  is  $\hat{\mu} = \frac{1}{N} \sum h(i_k)\pi(i_k)/p(i_k)$ . For the sake of estimating sum,  $\pi(i) = 1$  and  $h(i)$  is the value of  $i$ th variable  $x_i$ . In linear weighted sampling,  $p(i) = x_i/S$ , where  $S$  is exactly the sum we are trying to estimate, therefore we are not able to compute this estimator  $\hat{\mu}$  for sum. Another estimator is

$$\tilde{\mu} = \frac{\sum h(i_k)\pi(i_k)/\tilde{p}(i_k)}{\sum \pi(i_k)/\tilde{p}(i_k)},$$

where  $\tilde{p}$  is identical to  $p$  up to normalization and thus computable. However, the variance of  $\tilde{\mu}$  is even larger than the variance using uniform sampling.

A related topic is priority sampling and threshold sampling for estimating subset sums proposed and analyzed in [9,11,17]. But their cost model and application are quite different: they aim at building a sketch so that the sum of any subset can be computed (approximately) by only looking at the sketch; in particular their cost is defined as the size of the sketch and they can read all variables for free, so computing the total sum is trivial in their setting.

There is extensive work in estimating other frequency moments  $F_k = \sum x_i^k$  (sum is the first moment  $F_1$ ), in the random sampling model as well as in the streaming model (see for example [2,8,5]). The connection between the two models is discussed in [5]. Note that their sampling primitive is different from ours, and they assume  $F_1$  is known.



## 2 Definitions and Summary of Results

Let  $x_1, x_2, \dots, x_n$  be  $n$  variables. We consider the problem of estimating the sum  $S = \sum_i x_i$ , given  $n$ . We also refer to variables as *buckets* and the value of a variable as its *bucket size*.

In (*general*) *weighted sampling* we can sample a bucket  $x_i$  with probability proportional to a function of its size  $f(x_i)$ , where  $f$  is an arbitrary function of  $x_i$  ( $f$  independent on  $n$ ). Two special cases are *uniform sampling* where each bucket is sampled uniformly at random ( $f(x) = 1$ ), and *linear weighted sampling* where the probability of sampling a bucket is proportional to its size ( $f(x) = x$ ). We assume sampling with replacement.

We say an algorithm is an  $(\epsilon, \delta)$ -*estimator* ( $0 < \epsilon, \delta < 1$ ), if it outputs an estimated sum  $S'$  such that with probability at least  $1 - \delta$ ,  $|S' - S| \leq \epsilon S$ . The algorithm can take random samples of the buckets using some sampling method and learn the sizes as well as the labels of the sampled buckets. We measure the complexity of the algorithm by the total number of samples it takes. The algorithm has no prior knowledge of the bucket size distribution.

The power of the sum estimator is constrained by the sampling methods it is allowed to use. This paper studies the upper and lower bounds of the complexities of  $(\epsilon, \delta)$ -estimators under various sampling methods. As pointed out in Section 1, using only uniform sampling there is no  $(\epsilon, \delta)$ -estimator with sub-linear samples.

First we show an  $(\epsilon, \delta)$ -estimator using linear weighted sampling with  $\tilde{O}(\sqrt{n})$  samples. While linear weighted sampling is a natural sampling method, to derive the sum from such samples does not seem straightforward. Our scheme first converts the general problem to a special case where all buckets are either empty or of a fixed size; now the problem becomes estimating the number of non-empty buckets and we make use of birthday paradox by examining how many samples are needed to find a repeat. Each step involves some non-trivial construction and the detailed proof is presented in Section 3.

In Section 4 we consider sum estimators where both uniform and linear weighted sampling are allowed. Section 4.1 proposes an algorithm with  $\tilde{O}(\sqrt[3]{n})$  samples which builds upon the linear weighted sampling algorithm in Section 3. Section 4.2 gives a different algorithm with  $\tilde{O}(\sqrt{n})$  samples: although it is asymptotically worse than the former algorithm in terms of  $n$ , it has better dependency on  $\epsilon$  and a much smaller hidden constant; also this algorithm is much neater and easier to implement.

Finally we present lower bounds in Section 5. We prove that the algorithms in Section 3 and 4.1 are almost optimal in terms of  $n$  up to a poly-log factor. More formally, we prove a lower bound of  $\Omega(\sqrt{n})$  samples using only linear weighted sampling (more generally, using any combination of general weighted sampling methods with the constraint  $f(0) = 0$ ); a lower bound of  $\Omega(\sqrt[3]{n})$  samples using any combination of general weighted sampling methods.

All algorithms and bounds can be extended to the case where the number of buckets  $n$  is only approximately known (with relative error less than  $\epsilon$ ). We omit the details for lack of space.

### 3 An $\tilde{O}(\sqrt{n})$ Estimator Using Linear Weighted Sampling

Linear weighted sampling is a natural sampling method, but to efficiently derive the sum from such samples does not seem straightforward. Our algorithm first converts the general problem to a special case where all buckets are either empty or of a fixed size, and then tackle the special case making use of the *birthday paradox*, which states that given a group of  $\sqrt{365}$  randomly chosen people, there is a good chance that at least two of them have the same birthday.

Let us first consider the special case where all non-zero buckets are of equal sizes. Now linear weighted sampling is equivalent to uniform sampling among non-empty buckets, and our goal becomes estimating the number of non-empty buckets, denoted by  $B$  ( $B \leq n$ ). We focus on a quantity we call “*birthday period*”, which is the number of buckets sampled until we see a repeated bucket. We denote by  $r(B)$  the birthday period of  $B$  buckets; its expected value  $E[r(B)]$  is  $\Theta(\sqrt{B})$  according to the birthday paradox. We will estimate the expected birthday period using linear weighted sampling, and then use it to infer the value of  $B$ . Most runs of birthday period take  $O(\sqrt{B}) = O(\sqrt{n})$  samples, and we can cut off runs which take too long;  $\lg \frac{1}{\delta}$  runs are needed to boost confidence, thus in total we need  $O(\sqrt{n})$  samples to estimate  $B$ .

Now back to the general problem. We first guess the sum is  $an$  and fix a uniform bucket size  $\epsilon a$ . For each bucket in the original problem, we round its size down to  $k\epsilon a$  ( $k$  being an integer) and break it into  $k$  buckets. If our guess of sum is (approximately) right, then the number of new buckets  $B$  is approximately  $n/\epsilon$ ; otherwise  $B$  is either too small or too large. We can estimate  $B$  by examining the birthday period as above using  $O(\sqrt{n/\epsilon})$  samples, and check whether our guess is correct. Finally, since we allow a multiplicative error of  $\epsilon$ , a logarithmic number of guesses suffice.

Before present the algorithm, we first establish some basic properties of birthday period  $r(B)$ . The following lemma bounds the expectation and variance of  $r(B)$ ; property (3) shows that birthday period is “gap preserving” so that if the number of buckets is off by an  $\epsilon$  factor, we will notice a difference of  $c\epsilon$  in the birthday period. We can write out the exact formula for  $E[r(B)]$  and  $\text{var}(r(B))$ , and the rest of the proof is merely algebraic manipulation. The detailed proof can be found in the full version of the paper [16].

**Lemma 1.** (1)  $E[r(B)]$  monotonically increases with  $B$ ;  
 (2)  $E[r(B)] = \Theta(\sqrt{B})$ ;  
 (3)  $E[r((1 + \epsilon)B)] > (1 + c\epsilon)E[r(B)]$ , where  $c$  is a constant.  
 (4)  $\text{var}(r(B)) = O(B)$ ;

Lemma 2 tackles the special case, stating that with  $\sqrt{b}$  samples we can tell whether the total number of buckets is at most  $b$  or at least  $b(1 + \epsilon)$ . The idea is to measure the birthday period and compare with the expected period in the two cases. We use the standard “median of the mean” trick: first get a constant correct probability using Chebyshev inequality, then boost the probability using Chernoff bound. See details in the algorithm *BucketNumber*. Here  $c$  is the constant in Lemma 1(3);  $c_1$  and  $c_2$  are constants.

---

BucketNumber( $b, \epsilon, \delta$ )

1. Compute  $r = E[r(b)]$ ;
  2. for  $i = 1$  to  $k_1 = c_1 \lg \frac{1}{\delta}$
  3.     for  $j = 1$  to  $k_2 = c_2/\epsilon^2$
  4.         sample until see a repeated bucket; let  $r_j$  be the number of samples
  5.         if  $\sum_{j=1}^{k_2} r_j/k_2 \leq (1 + c\epsilon/2)r$  then  $s_i = \text{true}$ , else  $s_i = \text{false}$
  6. if more than half of  $s_i$  are *true* then output “ $\leq b$  buckets”  
     else output “ $\geq b(1 + \epsilon)$  buckets”
- 

**Lemma 2.** *If each sample returns one of  $B$  buckets uniformly at random, then the algorithm BucketNumber tells whether  $B \leq b$  or  $B \geq b(1 + \epsilon)$  correctly with probability at least  $1 - \delta$ ; it uses  $\Theta(\sqrt{b} \lg \frac{1}{\delta} / \epsilon^2)$  samples.*

*Proof.* We say the algorithm does  $k_1$  runs, each run consisting of  $k_2$  iterations. We first analyze the complexity of the algorithm. We need one small trick to avoid long runs: notice that we can cut off a run and set  $s_i = \text{false}$  if we have already taken  $(1 + c\epsilon/2)rk_2$  samples in this run. Therefore the total number of samples is at most

$$(1 + c\epsilon/2)rk_2k_1 = (1 + c\epsilon/2)E[r(b)]\frac{c_2}{\epsilon^2}c_1 \lg \frac{1}{\delta} = \Theta\left(\frac{\sqrt{b} \lg \frac{1}{\delta}}{\epsilon^2}\right).$$

The last equation uses Property (2) of Lemma [1](#).

Below we prove the correctness of the algorithm. Consider one of the  $k_1$  runs. Let  $r'$  be the average of the  $k_2$  measured birthday periods  $r_j$ . Because each measured period has mean  $E[r(B)]$  and variance  $\text{var}(r(B))$ , we have  $E[r'] = E[r(B)]$  and  $\text{var}(r') = \text{var}(r(B))/k_2$ .

If  $B \leq b$ , then  $E[r'] = E[r(B)] \leq r$ . By Chebyshev inequality [15](#),

$$Pr[r' > (1 + \frac{c\epsilon}{2})r] \leq Pr[r' > E[r(B)] + \frac{rc\epsilon}{2}] \leq \frac{\text{var}(r(B))/k_2}{(rc\epsilon/2)^2} \leq \frac{O(b)\epsilon^2/c_2}{(\Theta(\sqrt{b})c\epsilon/2)^2} = \frac{O(1)}{c_2}$$

If  $B \geq b(1 + \epsilon)$ , then  $E[r'] \geq E[r(b(1 + \epsilon))] \geq (1 + c\epsilon)r$  by Lemma [1](#).

$$Pr[r' < (1 + \frac{c\epsilon}{2})r] \leq Pr[r' < (1 - \frac{c\epsilon}{4})E[r']] \leq \frac{\text{var}(r(B))/k_2}{(E[r(B)]c\epsilon/4)^2} = \frac{O(1)}{c_2}$$

We choose the constant  $c_2$  large enough such that both probabilities are no more than  $1/3$ . Now when  $B \leq b$ , since  $Pr[r' > (1 + c\epsilon/2)r] \leq 1/3$ , each run sets  $s_i = \text{false}$  with probability at most  $1/3$ . Our algorithm makes wrong judgement only if more than half of the  $k_1$  runs set  $s_i = \text{false}$ , and by Chernoff bound [15](#), this probability is at most  $e^{-c'k_1}$ . Choose appropriate  $c_1$  so that the error probability is at most  $\delta$ . Similarly, when  $B \geq (1 + \epsilon)b$ , each run sets  $s_i = \text{true}$  with probability at most  $1/3$ , and the error probability of the algorithm is at most  $\delta$ .  $\square$

Algorithm *LWSE* (stands for *Linear Weighted Sampling Estimator*) shows how to estimate sum for the general case. The labelling in step 3 is equivalent to the following process: for each original bucket, round its size down to a multiple

of  $\epsilon_1 a$  and split into several “standard” buckets each of size  $\epsilon_1 a$ ; each time sampling returns a standard bucket uniformly at random. The two processes are equivalent because they have the same number of distinct labels (standard buckets) and each sampling returns a label uniformly at random. Therefore by calling  $\text{BucketNumber}(n(1 + \epsilon_1)/\epsilon_1, \epsilon_1, \delta_1)$  with such samples, we can check whether the number of standard buckets  $B \leq n(1 + \epsilon_1)/\epsilon_1$  or  $B \geq n(1 + \epsilon_1)^2/\epsilon_1$ , allowing an error probability of  $\delta_1$ .

---

LWSE( $n, \epsilon, \delta$ )

1. get a lower bound  $L$  of the sum: sample one bucket using linear weighted sampling and let  $L$  be the size of the sampled bucket;
  2. for  $a = L/n, L(1 + \epsilon_1)/n, \dots, L(1 + \epsilon_1)^k/n, \dots$  (let  $\epsilon_1 = \epsilon/3$ )
  3. for each sample returned by linear weighted sampling, create a label as follows: suppose a bucket  $x_i$  of size  $s = m\epsilon_1 a + r$  is sampled ( $m$  is an integer and  $r < \epsilon_1 a$ ); discard the sample with probability  $r/s$ ; with the remaining probability generate a number  $l$  from  $1..m$  uniformly at random and label the sample as  $li$ ;
  4. call  $\text{BucketNumber}(n(1 + \epsilon_1)/\epsilon_1, \epsilon_1, \delta_1)$ , using the above samples in step 4 of  $\text{BucketNumber}$ . If  $\text{BucketNumber}$  outputs “ $\leq n(1 + \epsilon_1)/\epsilon_1$ ”, then output  $S' = an$  as the estimated sum and terminate.
- 

**Theorem 1.** *LWSE is an  $(\epsilon, \delta)$ -estimator with  $O(\sqrt{n}(\frac{1}{\epsilon})^{\frac{5}{2}} \log n(\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log \log n))$  samples, where  $n$  is the number of buckets.*

*Proof.* We first show that the algorithm terminates with probability at least  $1 - \delta_1$ .  $S$  must fall in  $[a_0 n, a_0 n(1 + \epsilon_1)]$  for some  $a_0$ , and we claim that the algorithm will terminate at this  $a_0$ , if not before: since  $S \leq a_0 n(1 + \epsilon_1)$ , the sum after rounding down is at most  $a_0 n(1 + \epsilon_1)$  and hence the number of standard buckets  $B \leq n(1 + \epsilon_1)/\epsilon_1$ ; by Lemma 2 it will pass the check with probability at least  $1 - \delta_1$  and terminate the algorithm.

Next we show that given that  $LWSE$  terminates by  $a_0$ , the estimated sum is within  $(1 \pm \epsilon)S$  with probability  $1 - \delta_1$ . Since the algorithm has terminated by  $a_0$ , the estimated sum cannot be larger than  $S$ , so the only error case is  $S' = an < (1 - \epsilon)S$ . The sum loses at most  $n\epsilon_1$  after rounding down, so

$$B \geq \frac{S - an\epsilon_1}{a\epsilon_1} \geq \frac{\frac{an}{1-\epsilon} - an\epsilon_1}{a\epsilon_1} = \frac{n}{(1-\epsilon)\epsilon_1} - n \geq n \frac{1-\epsilon_1}{(1-\epsilon)\epsilon_1} \geq n \frac{(1+\epsilon_1)^2}{\epsilon_1}$$

The probability that it can pass the check for a fixed  $a < a_0$  is at most  $\delta_1$ ; by union bound, the probability that it passes the check for any  $a < a_0$  is at most  $\delta_1 \log_{1+\epsilon} \frac{S}{L}$ . Combining the two errors, the total error probability is at most  $\delta_1(\log_{1+\epsilon} \frac{S}{L} + 1)$ . Choose  $\delta_1 = \delta/(\log_{1+\epsilon} \frac{S}{L} + 1)$ , then with probability at least  $1 - \delta$  the estimator outputs an estimated sum within  $(1 \pm \epsilon)S$ .

Now we analyze the complexity of  $LWSE$ . Ignore the discarded samples for now and count the number of valid samples. By Lemma 2, for each  $a$  we need

$$N_1 = O\left(\frac{\log \frac{1}{\delta_1} * \sqrt{\frac{n(1+\epsilon_1)}{\epsilon_1}}}{\epsilon_1^2}\right) = O\left(\sqrt{n}\left(\frac{1}{\epsilon}\right)^{\frac{5}{2}}\left(\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log \log \frac{S}{L}\right)\right)$$

samples, and there are  $\log_{1+\epsilon} \frac{S}{L} = O(\log \frac{S}{L}/\epsilon)$  as. As for the discarded samples, the total discarded size is at most  $an\epsilon_1$ , and we always have  $S \geq an$  if the algorithm is running correctly, therefore the expected probability of discarded samples is at most  $\epsilon_1 = \epsilon/3 \leq 1/3$ . By Chernoff bound, with high probability the observed probability of discarded samples is at most half, i.e. the discarded samples at most add a constant factor to the total sample number.

Finally, the complexity of the estimator has the term  $\log \frac{S}{L}$ . Had we simply started guessing from  $L = 1$ , the cost would depend on  $\log S$ . The algorithm chooses  $L$  to be the size of a sampled bucket using linear weighted sampling. We claim that with high probability  $L \geq S/n^2$ : otherwise  $L < S/n^2$ , then the probability that linear weighted sampling returns any bucket of size no more than  $L$  is at most  $n * L/S < 1/n$ .

Summing up, the total sample number used in *LWSE* is

$$N_1 * O\left(\frac{\log n^2}{\epsilon}\right) = O\left(\sqrt{n}\left(\frac{1}{\epsilon}\right)^{\frac{2}{3}} \log n \left(\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log \log n\right)\right). \quad \square$$

## 4 Combining Uniform and Linear Weighted Sampling

In this section we design sum estimators using both uniform sampling and linear weighted sampling. We present two algorithms. The first algorithm uses *LWSE* in Section 3 as a building block and only needs  $\tilde{O}(\sqrt[3]{n})$  samples. The second algorithm is self-contained and easier to implement; its complexity is worse than the first algorithm in terms of  $n$  but has better dependency on  $\epsilon$  and a much smaller hidden constant.

### 4.1 An Estimator with $\tilde{O}(\sqrt[3]{n})$ Samples

In this algorithm, we split the buckets into two types:  $\Theta(\sqrt[3]{n^2})$  *large* buckets and the remaining *small* buckets. We estimate the partial sum of the large buckets using linear weighted sampling as in Section 3; we stratify the small buckets into different size ranges and estimate the number of buckets in each range using uniform sampling.

**Theorem 2.** *CombEst is an  $(\epsilon, \delta)$ -estimator with  $O(n^{1/3}(\frac{1}{\epsilon})^{\frac{2}{3}} \log n (\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log \log n))$  samples, where  $n$  is the number of buckets.*

*Proof.* We analyze the error of the estimator. Denote by  $S_{large}(S_{small})$  the actual total size of large (small) buckets; by  $n_i$  the actual bucket number in level  $i$ .

In Step 2, since we are using linear weighted sampling, the expected fraction of large buckets in the samples equals to  $S_{large}/S$ . If  $S_{large}/S > \epsilon_1$ , then by Chernoff bound the observed fraction of large buckets in the sample is larger than  $\epsilon_1/2$  with high probability, and we will get  $S'_{large}$  within  $(1 \pm \epsilon_1)S_{large}$  with probability at least  $1 - \delta/2$  according to Theorem 1; otherwise we lose at most  $S_{large} = \epsilon_1 S$  by estimating  $S'_{large} = 0$ . Thus, with probability at least  $1 - \delta/2$ , the error introduced in Step 2 is at most  $\epsilon_1 S$ .

---

 CombEst( $n, \epsilon, \delta$ )

1. find  $t$  such that the number of buckets whose sizes are larger than  $t$  is  $N_t = \Theta(n^{2/3})$  (we leave the detail of this step later); call a bucket *large* if its size is above  $t$ , and *small* otherwise
  2. use linear weighted sampling to estimate the total size of large buckets  $S'_{large}$ : if the fraction of large buckets in the sample is less than  $\epsilon_1/2$ , let  $S'_{large} = 0$ ; otherwise ignore small buckets in the samples and estimate  $S'_{large}$  using  $LWSE(N_t, \epsilon_1, \delta/2)$ , where  $\epsilon_1 = \epsilon/4$
  3. use uniform sampling to estimate the total size of small buckets  $S'_{small}$ : divide the small bucket sizes into levels  $[1, 1 + \epsilon_1), \dots, [(1 + \epsilon_1)^i, (1 + \epsilon_1)^{i+1}), \dots, [(1 + \epsilon_1)^{i_0}, t)$ ; we say a bucket in level  $i$  ( $0 \leq i \leq i_0$ ) if its size  $\in [(1 + \epsilon_1)^i, (1 + \epsilon_1)^{i+1})$  make  $k = \Theta(n^{1/3} \log n / \epsilon_1^4)$  samples using uniform sampling; let  $k_i$  be the number of sampled buckets in level  $i$ . Estimate the total number of buckets in level  $i$  to be  $n'_i = k_i n / k$  and  $S'_{small} = \sum_i n'_i (1 + \epsilon_1)^i$
  4. output  $S'_{small} + S'_{large}$  as the estimated sum
- 

In Step 3, it is easy to see that  $n'_i$  is an unbiased estimator of  $n_i$ . For a fixed  $i$ , if  $n_i \geq \epsilon_1^2 n^{2/3}$  then by Chernoff bound the probability that  $n'_i$  deviates from  $n_i$  by more than an  $\epsilon_1$  fraction is

$$Pr[|n'_i - n_i| \geq \epsilon_1 n_i] \leq \exp(-c k \epsilon_1^2 n_i / n) \leq \exp(-c' \frac{n^{1/3} \log n}{\epsilon_1^4} \epsilon_1^2 \frac{\epsilon_1^2 n^{2/3}}{n}) = n^{-c'}$$

This means that for all  $n_i \geq \epsilon_1 n^{2/3}$ , with high probability we estimate  $n_i$  almost correctly, introducing a relative error of at most  $\epsilon_1$ .

We round all bucket sizes of small buckets down to the closest power of  $1 + \epsilon_1$ ; this rounding introduces a relative error of at most  $\epsilon_1$ .

For all levels with  $n_i < \epsilon_1^2 n^{2/3}$ , the total bucket size in those levels is at most

$$\sum_{0 \leq i \leq i_0} n_i (1 + \epsilon_1)^{i+1} < \epsilon_1^2 n^{2/3} \sum_i (1 + \epsilon_1)^{i+1} < \epsilon_1^2 n^{2/3} \frac{t}{\epsilon_1} = \epsilon_1 t n^{2/3} < \epsilon_1 S_{large} < \epsilon_1 S$$

The errors introduced by those levels add up to at most  $\epsilon_1$ .

Summing up, there are four types of errors in our estimated sum, with probability at least  $1 - \delta$  each contributing at most  $\epsilon_1 S = \epsilon S/4$ , so  $S'$  has an error of at most  $\epsilon S$ .

Now we count the total number of samples in *CombEst*. According to Theorem [1](#), Step 2 needs  $O(\sqrt{n^{2/3}} (\frac{1}{\epsilon})^{\frac{7}{2}} \log n^{2/3} (\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log \log n^{2/3}))$  samples of large buckets, and by our algorithm the fraction of large buckets is at least  $\epsilon_1/2$ . Step 3 needs  $\Theta(n^{1/3} \log n / \epsilon_1^4)$  samples, which is dominated by the sample number of Step 2. Therefore the total sample number is

$$O(n^{1/3} (\frac{1}{\epsilon})^{\frac{9}{2}} \log n (\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log \log n)). \quad \square$$

There remains to be addressed the implementation of Step 1. We make  $n^{1/3} \log n$  samples using uniform sampling and let  $t$  be the size of the  $2 \log n$ -th largest bucket in the samples. If all sampled buckets have different sizes, then we can

show that with high probability  $n^{2/3} \leq N_t \leq 4n^{2/3}$ . Details and handling of ties in the full version [16]. Finally we only know the approximate number of large buckets, denoted by  $N'_t$ , and have to pass  $N'_t$  instead of  $N_t$  when call *LWSE*. Fortunately an approximate count of  $n$  suffices for *LWSE*, and a constant factor error in  $n$  only adds a constant factor in the complexity.

## 4.2 An Estimator with $\tilde{O}(\sqrt{n})$ Samples

Next we present a sum estimator using uniform and weighted sampling with  $\tilde{O}(\sqrt{n})$  samples. Recall that uniform sampling works poorly for skewed distributions, especially when there are a few large buckets that we cannot afford to miss. The idea of this algorithm is to use weighted sampling to deal with such heavy tails: if a bucket is large enough it will keep appearing in weighted sampling; after enough samples we can get a fairly accurate estimate of its frequency of being sampled, and then infer the total size by only looking at the size and sampling frequency of this bucket. On the other hand, if no such large bucket exists, the variance cannot be too large and uniform sampling performs well.

---

CombEstSimple( $n, \epsilon, \delta$ )

1. Make  $k = c_1 \sqrt{n} \log \frac{1}{\delta} / \epsilon^2$  samples using linear weighted sampling. Suppose the most frequently sampled bucket has size  $t$  and is sampled  $k_1$  times (breaking ties arbitrarily). If  $k_1 \geq k/2\sqrt{n}$ , output  $S' = tk/k_1$  as estimated sum and terminate.
  2. Make  $l = \sqrt{n}/\delta\epsilon^2$  samples using uniform sampling and let  $a$  be the average of sampled bucket sizes. Output  $S' = an$  as estimated sum.
- 

**Theorem 3.** *CombEstSimple is an  $(\epsilon, \delta)$ -estimator with  $O(\sqrt{n}/\epsilon^2\delta)$  samples.*

We present the sketch of the proof for lack of space. The detailed proof can be found in the full version [16].

*Proof sketch.* Obviously *CombEstSimple* uses  $k+l = O(\sqrt{n}/\epsilon^2\delta)$  samples. Below we analyze the accuracy of the estimator.

We first prove that if Step 1 outputs an estimated sum  $S'$ , then  $S'$  is within  $(1 \pm \epsilon)S$  with probability  $1 - \delta/2$ . Consider any bucket with size  $t$  whose frequency of being sampled  $f' = k_1/k$  is more than  $1/2\sqrt{n}$ . Its expected frequency of being sampled is  $f = t/S$ , so we can bound the error  $|f' - f|$  using Chernoff bound:  $Pr[|f - f'| > \epsilon f] \leq \delta^{\Theta(c_1)}$ . Choose  $c_1$  large enough to make  $Pr[|f - f'| > \epsilon f]$  less than  $\delta/2$ , then with probability  $1 - \delta/2$ ,  $f' = k_1/k$  is within  $(1 \pm \epsilon)t/S$ , and it follows that the estimated sum  $tk/k_1$  is within  $(1 \pm \epsilon)S$ .

If Step 1 does not output an estimated sum, then again using Chernoff bound we can show that the maximum bucket size is no more than  $S/\sqrt{n}$  with probability at most  $\delta/2$ . In this case, the statistical variance of  $x_i$  is

$$\text{var}(x) \leq E[x^2] = \frac{\sum_i x_i^2}{n} \leq \frac{(\frac{S}{\sqrt{n}})^2 \sqrt{n}}{n} = \frac{S^2}{n\sqrt{n}}$$

$a$  is an unbiased estimator of the mean bucket size, and its variance is  $\text{var}(x)/l$ . Using Chebyshev inequality, the probability that  $a$  deviates from the actual mean  $S/n$  by more than an  $\epsilon$  fraction is at most  $\text{var}(a)/(\epsilon S/n)^2 = \sqrt{n}/l\epsilon^2 = \delta$ .  $\square$

## 5 Lower Bounds

Finally we prove lower bounds on the sample number of sum estimators. Those lower bound results use a special type of input instances where all bucket sizes are either 0 or 1. The results still hold if all bucket sizes are strictly positive, using similar counterexamples with bucket sizes either 1 or a large constant  $b$ .

**Theorem 4.** *There exists no  $(\epsilon, \delta)$ -estimator with  $o(\sqrt{n})$  samples using only linear weighted sampling, for any  $0 < \epsilon, \delta < 1$ .*

*Proof.* Consider two instances of inputs: in one input all buckets have size 1; in the other,  $(1 - \epsilon)n/(1 + \epsilon)$  buckets have size 1 and the remaining are empty. If we cannot distinguish the two inputs, then the estimated sum deviates from the actual sum by more than an  $\epsilon$  fraction. For those two instances, linear weighted sampling is equivalent to uniform sampling among non-empty buckets. If we sample  $k = o(\sqrt{n})$  buckets, then the probability that we see a repeated bucket is less than  $1 - \exp(-k(k - 1)/((1 - \epsilon)n/(1 + \epsilon))) = o(1)$  (see the proof of Lemma [11](#)). Thus in both cases with high probability we see all distinct buckets of the same sizes, so cannot distinguish the two inputs in  $o(\sqrt{n})$  samples.  $\square$

More generally, there is no estimator with  $o(\sqrt{n})$  samples using any combination of general weighted sampling methods with the constraint  $f(0) = 0$ . Recall that weighted sampling with function  $f$  samples a bucket  $x_i$  with probability proportional to a function of its size  $f(x_i)$ . When  $f(0) = 0$ , it samples any empty bucket with probability 0 and any bucket of size 1 with the same probability, thus is equivalent to linear weighted sampling for the above counterexample.

**Theorem 5.** *There exists no  $(\epsilon, \delta)$ -estimator with  $o(\sqrt[3]{n})$  samples using any combination of general weighted sampling (the sampling function  $f$  independent on  $n$ ), for any  $0 < \epsilon, \delta < 1$ .*

*Proof.* Consider two instances of inputs: in one input  $n^{2/3}$  buckets have size 1 and the remaining buckets are empty; in the other,  $3n^{2/3}$  buckets have size 1 and the remaining are empty. If we cannot distinguish the two inputs, then the estimated sum deviates from the actual sum by more than  $\frac{1}{2}$ . We can adjust the constant to prove for any constant  $\epsilon$ .

We divide weighted sampling into two types:

(1)  $f(0) = 0$ . It samples any empty bucket with probability 0 and any bucket of size 1 with the same probability, thus it is equivalent to uniform sampling among non-empty buckets. There are at least  $n^{2/3}$  non-empty buckets and we only make  $o(n^{1/3})$  samples, with high probability we see  $o(n^{1/3})$  distinct buckets of size 1 for both inputs.

(2)  $f(0) > 0$ . The probability that we sample any non-empty buckets is

$$\frac{f(1)cn^{2/3}}{f(1)cn^{2/3} + f(0)(n - cn^{2/3})} = \Theta(n^{-1/3}),$$



so in  $o(n^{1/3})$  samples with high probability we only see empty buckets for both inputs, and all these buckets are distinct.

Therefore whatever  $f$  we choose, we see the same sampling results for both inputs in the first  $o(n^{1/3})$  samples, i.e. we cannot distinguish the two inputs with  $o(n^{1/3})$  samples using any combination of weighted sampling methods.  $\square$

## Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable comments, especially for pointing out an implicit assumption in the proof.

## References

1. Alon, N., Duffield, N.G., Lund, C., Thorup, M.: Estimating arbitrary subset sums with few probes. In: PODS 2005
2. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. JCS 58, 137–147 (1999)
3. Bar-Yossef, Z., Gurevich, M. (eds.): Random sampling from a search engine’s index. In: WWW 2006
4. Bar-Yossef, Z., Gurevich, M.: Efficient search engine measurements. In: WWW 2007
5. Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Sampling algorithms: lower bounds and applications. In: STOC 2001
6. Broder, A., Fontura, M., Josifovski, V., Kumar, R., Motwani, R., Nabar, S., Panigrahy, R., Tomkins, A., Xu, Y.: Estimating corpus size via queries. In: CIKM 2006
7. Canetti, R., Even, G., Goldreich, O.: Lower Bounds for Sampling Algorithms for Estimating the Average. Information Processing Letters 53, 17–25 (1995)
8. Charikar, M., Chaudhuri, S., Motwani, R., Narasayya, V.: Towards estimation error guarantees for distinct values. In: PODS 2000
9. Duffield, N.G., Lund, C., Thorup, M.: Learn more, sample less: control of volume and variance in network measurements. IEEE Trans. on Information Theory 51, 1756–1775 (2005)
10. Gulli, A., Signorini, A.: The indexable Web is more than 11.5 billion pages. In: WWW 2005
11. Henzinger, M.R., Heydon, A., Mitzenmacher, M., Najork, M.: On near-uniform URL sampling. In: WWW 2000
12. Lawrence, S., Giles, C.: Searching the World Wide Web. Science 280, 98–100 (1998)
13. Lawrence, S., Giles, C.: Accessibility of information on the web. Nature 400, 107–109 (1999)
14. Liu, J.: Metropolized independent sampling with comparisons to rejection sampling and importance sampling. Statist. Comput. 6, 113–119 (1996)
15. Motwani, R., Raghavan, P.: Randomized Algorithm (1995)
16. Motwani, R., Raghavan, P., Xu, Y.: Estimating Sum by Weighted Sampling. Technical Report (2007)
17. Szegedy, M.: The DLT priority sampling is essentially optimal. In: STOC 2006

# Sampling Methods for Shortest Vectors, Closest Vectors and Successive Minima

Johannes Blömer\* and Stefanie Naewe\*\*

Department of Computer Science, University of Paderborn  
{bloemer,naestef}@uni-paderborn.de

**Abstract.** In this paper we introduce a new lattice problem, the *subspace avoiding problem* (SAP). We describe a probabilistic single exponential time algorithm for SAP for arbitrary  $\ell_p$  norms. We also describe polynomial time reductions for four classical problems from the geometry of numbers, the *shortest vector problem* (SVP), the *closest vector problem* (CVP), the *successive minima problem* (SMP), and the *shortest independent vectors problem* (SIVP) to SAP, establishing probabilistic single exponential time algorithms for them. The result generalize and extend previous results of Ajtai, Kumar and Sivakumar. The results on SMP and SIVP are new for all norms. The results on SVP and CVP generalize previous results of Ajtai et al. for the  $\ell_2$  norm to arbitrary  $\ell_p$  norms.

## 1 Introduction

In this paper we study four problems from the geometry of numbers, the *shortest vector problem* (SVP), the *closest vector problem* (CVP), the *successive minima problem* (SMP) and the *shortest linearly independent vectors problem* (SIVP).

In the shortest vector problem, we are given a lattice  $L$  and are asked to find a (almost) shortest non-zero vector  $v$  in the lattice  $L$ . In the closest vector problem, we are given a lattice  $L$  and some vector  $t$  in the  $\mathbb{R}$ -vector space  $\text{span}(L)$  spanned by the vectors in  $L$ . We are asked to find a vector  $u \in L$ , whose distance to  $t$  is as small as possible. The problems SMP and SIVP extend SVP and deal with the successive minima  $\lambda_k(L)$  of a lattice. Let  $k$  be an integer less than or equal to the dimension of  $\text{span}(L)$  (called the rank of  $L$ ). The  $k$ -th successive minimum  $\lambda_k(L)$  of  $L$  is the smallest real number  $r$  such that  $L$  contains  $k$  linearly independent vectors of length at most  $r$ . In the successive minima problem SMP we are given a lattice  $L$  with rank  $n$ . We are asked to find  $n$  linearly independent vectors  $v_1, \dots, v_n$  such that the length of  $v_k, k = 1, \dots, n$ , is at most  $\lambda_k(L)$ . In SIVP we are asked to find  $n$  linearly independent vectors  $v_1, \dots, v_n$  such that the length of  $v_k$  is at most  $\lambda_n(L)$ . Clearly, SIVP is polynomial time reducible to SMP. Since

---

\* This research was supported by Deutsche Forschungsgemeinschaft, grant BL 314/5.

\*\* This research was partially supported by German Science Foundation (DFG), grant BL 314/5, and Research Training Group GK-693 of the Paderborn Institute for Scientific Computation (PaSCo).

they can be defined for any norm on  $\mathbb{R}^n$ , we stated these problems without referring to a specific norm.

*Algorithms for lattice problems.* In the last 25 years the complexity of the lattice problems SVP, CVP, SMP, and SIVP has been studied intensively. For the history of this problems we refer to [MG02]. It is known that all problems are NP-hard and even hard to approximate (see for example [Ajt98], [Mic00], [Kho05], [DKRS03], [BS99]). Let us briefly review the best algorithms for lattice problems that predate the results by Ajtai et al. The best algorithm to solve SVP optimally was due to Kannan [Kan87b]. Kannan's algorithm has a running time of  $n^{n/2}b^{\mathcal{O}(1)}$ , where  $n$  is the rank of the lattice  $L$  and  $b$  is its representation size, i.e., the number of bits used to describe the basis defining  $L$ . For CVP the best algorithm that optimally solves the problem was due to [Blö00]. It has a running time of  $n!b^{\mathcal{O}(1)}$ . Finally, the best deterministic algorithms for SMP and SIVP were also due to [Blö00]. Their running time is  $3^b n!b^{\mathcal{O}(1)}$ .

Of course, the best deterministic polynomial time algorithms for approximating all four lattice problems are based on the LLL-algorithm (see [LLL82]) and achieve single exponential approximation factors (see for example [Sch94], [Bab86], [Sch87] and [Kan87a]).

*The AKS results for SVP and CVP.* In a breakthrough paper [AKS01] Ajtai, Kumar, and Sivakumar describe a probabilistic algorithm that solves SVP optimally with probability exponentially close to 1. More precisely, the running time of their algorithm is  $(2^{nb})^{\mathcal{O}(1)}$ , i.e., single exponential only in the rank of the lattice. The AKS-algorithm is based on a novel sampling technique that generates short vectors from the input lattice  $L$ . Later, Ajtai, Kumar, and Sivakumar [AKS02] extended their sampling technique to solve CVP with approximation factor  $(1+\epsilon)$  for any  $\epsilon > 0$ . The running time of their algorithm is  $(2^{(1+1/\epsilon)nb})^{\mathcal{O}(1)}$ .

*Our contributions.* In this paper, we consider a variant of the AKS-sampling procedure (according to [AKS01] proposed by M. Sudan, described in lecture notes by O. Regev [Reg04]).

- We describe a general sampling procedure to compute short lattice vectors outside some given subspace. We call this the *subspace avoiding problem* (SAP).
- We show polynomial time reductions from exact and approximate versions of SAP to exact and approximate versions of SVP, CVP, SMP and SIVP.
- In consequence, we obtain single exponential time  $(1 + \epsilon)$  approximation algorithms for SVP, CVP, SMP and SIVP for all  $\ell_p$  norms. The running time is  $((2 + 1/\epsilon)^{nb})^{\mathcal{O}(1)}$ .
- By slightly modifying the sampling procedure and its analysis we are able to solve SAP provided there do not exist too many short lattice vectors outside the given subspace. As a consequence, we obtain single exponential time algorithms for SVP and for restricted versions of CVP and SIVP.

*Organization.* The paper is organized as follows. In Section 2 we state the most important facts used in this paper. In Section 3 we formally define the lattice problem SAP and prove polynomial time reductions from SVP, CVP, SMP, and SIVP to SAP. In Section 4 we show that the problem SAP can be approximated with factor  $1 + \epsilon$ ,  $\epsilon > 0$  arbitrary, by a sampling procedure. Finally, the modified sampling procedure solving restricted versions of SAP optimally is presented in Section 5.

## 2 Basic Definitions and Facts

For  $m > 0$  is  $\mathbb{R}^m$  a  $m$ -dimensional vector space over  $\mathbb{R}$ . The  $\ell_p$  norm of a vector  $x \in \mathbb{R}^m$  is defined by  $\|x\|_p = (\sum_{i=1}^m x_i^p)^{1/p}$  for  $1 \leq p < \infty$  and  $\|x\|_\infty = \max\{|x_i|, i = 1, \dots, m\}$  for  $p = \infty$ . In the sequel we consider the  $\ell_p$  norm for an arbitrary  $p$  with  $1 \leq p \leq \infty$ . We set  $B^{(p)}(x, r) := \{y \in \mathbb{R}^m \mid \|y - x\|_p < r\}$ . The volume  $\text{vol}(B^{(p)}(x, r))$  satisfies:

$$\text{For all } c > 0 \quad \text{vol}(B^{(p)}(x, c \cdot r)) = c^m \cdot \text{vol}(B^{(p)}(x, r)). \quad (1)$$

A lattice  $L$  is a discrete additive subgroup of  $\mathbb{R}^m$ . Each lattice  $L$  has a basis, i. e. a sequence  $b_1, \dots, b_n$  of  $n$  elements of  $L$  that generate  $L$  as an abelian group. We denote this by  $L = \mathcal{L}(b_1, \dots, b_n)$ . We call  $n$  the rank of  $L$ . If  $m = n$ , the lattice is full dimensional. In the rest of the paper we only consider full dimensional lattices. However, our results can easily be generalized to arbitrary lattices. For a basis  $B = \{b_1, \dots, b_n\}$  we define the half open parallelepiped  $\mathcal{P}(B) := \{\sum_{j=1}^n \alpha_j b_j \mid 0 \leq \alpha_j < 1, j = 1, \dots, n\}$ . For every vector  $v \in \mathbb{R}^n$  there is a unique representation  $v = u + w$  with  $u \in L$  and  $w \in \mathcal{P}(B)$ . We write  $v \equiv w \pmod L$ .

We always assume that  $L \subseteq \mathbb{Q}^n$ . The representation size  $b$  of a lattice  $L \subseteq \mathbb{Q}^n$  with respect to the basis  $\{b_1, \dots, b_n\}$  is the maximum of  $n$  and the binary lengths of the numerators and denominators of the coordinates of the basis vectors  $b_j$ . The representation size of a subspace  $M$  and the representation size of a vector  $u = \sum_{i=1}^n u_i b_i$  with  $u_i \in \mathbb{Q}$  with respect to  $\{b_1, \dots, b_n\}$  are defined in the same way. In the sequel, if we speak of the representation size of a lattice  $L$ , a subspace  $M$  or of a vector  $u \in \text{span}(L)$  without referring to some specific basis, we implicitly assume that some basis is given.

## 3 The Subspace Avoiding Problem SAP, Main Result, and Reductions for SVP, SMP, SIVP and CVP

**Definition 1.** *Given a lattice  $L$  and some subspace  $M \subset \text{span}(L)$ , we call the problem to compute a vector  $v \in L \setminus M$ , that is as short as possible, the subspace avoiding problem (SAP). We set*

$$\lambda_M^{(p)}(L) := \min\{r \in \mathbb{R} \mid \exists v \in L \setminus M, \|v\|_p \leq r\}.$$

It is not hard to show that for an LLL-reduced basis  $\{b_1, \dots, b_n\}$  we have  $\|b_k\|_2 \leq 2^{n-1} \lambda_M^{(2)}(L)$ , where  $k = \min\{1 \leq j \leq n \mid b_j \in L \setminus M\}$ . Therefore, we get

**Theorem 1.** *The LLL-algorithm can be used to approximate in polynomial time SAP for the  $\ell_2$  norm with factor  $2^{n-1}$ .*

In Section 4 we will show that in single exponential time we can approximate SAP with any factor  $1 + \epsilon$ ,  $0 < \epsilon \leq 2$ . More precisely

**Theorem 2.** *For all  $\ell_p$  norms,  $1 \leq p \leq \infty$ , there exists a randomized algorithm, that approximates SAP with probability exponentially close to 1. The approximation factor is  $1 + \epsilon$  for any  $0 < \epsilon \leq 2$  and the running time of the algorithm is  $((2 + 1/\epsilon)^n \cdot b)^{O(1)}$ , where  $b$  is the size of the lattice and the subspace.*

In the remainder of this section we show that there are polynomial time reductions from SVP, CVP, SMP and SIVP to SAP. Together with Theorem 2 this implies single exponential time approximation algorithms for SVP, CVP, SMP and SIVP. The core of the reductions is a suitable definition of the subspace.

For the reduction of SVP to SAP we choose  $M := \{0\} \subseteq \text{span}(L)$ . If we compute a (almost) shortest non-zero lattice vector  $u \in L \setminus M$ , we compute a (almost) shortest non-zero lattice vector  $u \in L$ .

**Theorem 3.** *For all  $\ell_p$  norms,  $1 \leq p \leq \infty$ , SVP with approximation factor  $1 + \epsilon$ ,  $\epsilon \geq 0$ , is polynomial time reducible to SAP with approximation factor  $1 + \epsilon$ .*

**Theorem 4.** *For all  $\ell_p$  norms,  $1 \leq p \leq \infty$ , SMP and SIVP with approximation factor  $1 + \epsilon$ ,  $\epsilon \geq 0$ , are polynomial time reducible to SAP with approximation factor  $1 + \epsilon$ .*

*Proof.* We are given access to an oracle  $\mathcal{A}$ , that solves SAP with an approximation factor  $1 + \epsilon$  for some arbitrary  $\epsilon \geq 0$ . Using this oracle we get a  $(1 + \epsilon)$ -approximation of the first successive minimum as in Theorem 3. For  $i > 1$  define  $M := \text{span}(v_1, \dots, v_{i-1})$  with  $v_1, \dots, v_{i-1} \in L$  linearly independent. Since  $\dim(M) < i$ , there exists a vector  $w \in L$  with  $\|w\|_p \leq \lambda_i^{(p)}(L)$  and  $w \notin M$ . Therefore,  $\lambda_M^{(p)}(L) \leq \lambda_i^{(p)}(L)$  and using the oracle  $\mathcal{A}$  with input  $L$  and  $M$  we get a  $(1 + \epsilon)$ -approximation for the  $i$ -th successive minimum.

The reduction of CVP to SAP relies on a lifting technique introduced by Kannan [Kan87b] and refined by Goldwasser and Micciancio [MG02] and Ajtai, Kumar and Sivakumar [AKS02]. A proof for it will be contained in the full version of this paper.

**Theorem 5.** *For all  $\ell_p$  norms,  $1 \leq p \leq \infty$ , the exact version of CVP is polynomial time reducible to the exact version of SAP. Also, for all  $\ell_p$  norms,  $1 \leq p \leq \infty$ , CVP with approximation factor  $(1 + \epsilon)(1 + \alpha)$  for  $0 < \epsilon \leq 1/2$  and  $\alpha \geq 0$  is reducible to SAP with approximation factor  $1 + \epsilon/6$ . The reduction is polynomial time in the input size of the CVP instance and in  $1/\alpha$ .*

Combining this result with the inapproximability results for CVP due to Dinur et al. [DKRS03] we get the following inapproximability result for SAP.

**Theorem 6.** *For all  $\ell_p$  norms,  $1 \leq p < \infty$ , there is some constant  $c > 0$ , such that SAP is NP-hard to approximate to within factor  $n^{c/\log \log n}$ , where  $n$  is the dimension of the input lattice.*

## 4 The Sieving Procedure and the Sampling Procedure

In this section we present a sampling procedure, that solves the subspace avoiding problem with approximation factor  $1 + \epsilon$ ,  $0 < \epsilon \leq 2$ . We closely follow Regev's lecture notes on the AKS single exponential algorithm for SVP [Reg04]. First, we show that we can restrict ourselves to instances of SAP with  $2 \leq \lambda_M^{(p)}(L) < 3$ . A proof for it will be contained in the full version of this paper.

**Lemma 1.** *For all  $\ell_p$  norms, if there is an algorithm  $\mathcal{A}$  that for all lattices  $L$  for which  $2 \leq \lambda_M^{(p)}(L) < 3$  and all subspaces  $M$  solves SAP with approximation factor  $1 + \epsilon$  and in time  $T = T(n, b, \epsilon)$ , then there is an algorithm  $\mathcal{A}'$  that solves SAP for all lattices and subspaces  $M$  with approximation factor  $1 + \epsilon$  and in time  $\mathcal{O}(nT + n^{4b})$ . Here  $n$  is the rank of  $L$  and  $b$  is the representation size of  $L, M$ .*

### 4.1 The Sieving Procedure

The main part of the sampling procedure is a sieving procedure (see Algorithm 2). Its main properties are described in the following lemma. The parameter  $a$  is rational and  $a > 1$ .

**Algorithm 2 The sieving procedure**

*Input:*  $x_1, \dots, x_N \in B^{(p)}(0, R)$

$J \leftarrow \emptyset$

For  $j = 1, \dots, N$  do

    If there exists  $i \in J$  with  $\|x_i - x_j\|_p \leq R/a$ , then  $\eta(i) \leftarrow j$ .

    Else  $J \leftarrow J \cup \{i\}$  and  $\eta(i) \leftarrow i$ .

**Lemma 2.** *Let  $R \in \mathbb{R}$ ,  $R > 0$ ,  $a \in \mathbb{Q}$  with  $a > 1$ . For any set of points  $x_1, \dots, x_N \in B^{(p)}(0, R)$  the sieving procedure 2 finds a subset  $J \subseteq \{1, 2, \dots, N\}$  of size at most  $(2a + 1)^n$  and a mapping  $\eta : \{1, 2, \dots, N\} \rightarrow J$  such that for any  $i \in \{1, \dots, N\}$ ,  $\|x_i - x_{\eta(i)}\|_p \leq R/a$ . The running time of the procedure is  $\mathcal{O}(N^2 \cdot \text{poly}(m))$ , if  $x_1, \dots, x_N$  are rationals of representation size  $m$ .*

*Proof.* Obviously, for all  $i \in \{1, \dots, N\}$ ,  $\|x_i - x_{\eta(i)}\|_p \leq R/a$ . The distance between any two points in  $J$  is larger than  $R/a$ . If we take balls of radius  $R/(2a)$  around each point  $x_i$ ,  $i \in J$ , then these balls are disjoint and their union is contained in  $B^{(p)}(0, (1 + 1/(2a))R)$ . Therefore, the number of balls, and hence  $|J|$ , is bounded by  $\text{vol}(B^{(p)}(0, (1 + \frac{1}{2a})R)) / \text{vol}(B^{(p)}(0, \frac{1}{2a}R)) = (2a + 1)^n$  (Equation (1)).

### 4.2 The Sampling Procedure

Now we present a sampling procedure (see Algorithm 3) that for all  $\ell_p$  norms approximates SAP with the factor  $1 + \epsilon$ ,  $0 \leq \epsilon \leq 2$  arbitrary. The algorithm chooses  $N$  points uniformly at random in a ball  $B^{(p)}(0, r)$  with radius  $r$ . Using the general algorithm of Dyer, Frieze and Kannan (see [DFK91]) we are able to

sample the  $N$  points in  $B^{(p)}(0, r)$  with the required accuracy. For the sake of simplicity, we will neglect this aspect in the following. The parameter  $N$  will be defined later. For each point  $x_i$  with  $i \in \{1, \dots, N\}$  we compute the point  $y_i \in \mathcal{P}(B)$  such that  $y_i - x_i$  is a lattice point. Using the mapping  $\eta : \{1, \dots, N\} \rightarrow J$ , for each vector  $y_i$  we get a representative  $y_{\eta(i)}$  with  $\|y_i - y_{\eta(i)}\|_p < R/a$ . We replace  $y_i$  with  $y_i - (y_{\eta(i)} - x_{\eta(i)})$ . This procedure is repeated until the distance between the lattice vectors and their representatives is small enough. We use parameters  $\delta$ ,  $r$  and  $a$  satisfying  $0 < \delta \leq 1/2$ ,  $r \geq 1/2$  and  $a = 1 + 2/\delta$ .

**Algorithm 3 The sampling procedure**

*Input:* A lattice  $L = \mathcal{L}(B)$ ,  $B = \{b_1, \dots, b_n\}$ , and a subspace  $M \subseteq \text{span}(B)$ .

1. (a)  $R_0 \leftarrow n \cdot \max_i \|b_i\|_p$   
 (b) Choose  $N$  points  $x_1, \dots, x_N$  uniformly in  $B^{(p)}(0, r)$ .  
 (c) Compute  $y_i \in \mathcal{P}(B)$  with  $y_i \equiv x_i \pmod{\mathcal{L}(B)}$  for  $i = 1, \dots, N$ .  
 (d) Set  $\mathcal{Z} \leftarrow \{(x_1, y_1), \dots, (x_N, y_N)\}$  and  $R \leftarrow R_0$ .
2. While  $R > (1 + \delta)r$  do
  - (a) Apply the sieving procedure to  $\{y_i \mid (x_i, y_i) \in \mathcal{Z}\}$  with the parameters  $a$  and  $R$ . The result is a set  $J$  and a mapping  $\eta$ .
  - (b) Remove from  $\mathcal{Z}$  all pairs  $(x_i, y_i)$  with  $i \in J$ .
  - (c) Replace each remaining pair  $(x_i, y_i) \in \mathcal{Z}$  with  $(x_i, y_i - (y_{\eta(i)} - x_{\eta(i)}))$ .
  - (d)  $R \leftarrow R/a + r$

*Output:* A shortest vector  $v \in \{y_i - x_i \mid (x_i, y_i) \in \mathcal{Z}\}$  with  $v \notin M$ , if such a vector exists.

**Lemma 3.** *Let  $\delta$  and  $r$  be chosen as above. Given a lattice  $L = \mathcal{L}(B)$ , if the sampling procedure [3](#) returns the vector  $v$ , then  $v \in L \cap B^{(p)}(0, (2 + \delta)r)$ .*

The proof follows from the fact that during the sampling procedure the following two properties are satisfied: 1) For all  $(x_i, y_i) \in \mathcal{Z}$  we have  $y_i - x_i \in \mathcal{L}(B)$  and 2) for all  $i \in \{1, \dots, N\}$  the length of  $y_i$  is bounded by the parameter  $R$ . For details see [Reg04](#).

The number of iterations of the while-loop dominates the running time of the sampling procedure.

**Lemma 4.** *If the sampling procedure [3](#) is executed with the parameters  $\delta$ ,  $a$  and  $r$  chosen as above, then the number of iterations of the while-loop is at most  $2 \log_2(1 + 2/\delta) \cdot (\log_2 R_0 + \log_2(1 + 2/\delta))$ .*

*Proof.* After  $i$  iterations the parameter  $R$  is  $R_0/a^i + r \sum_{j=0}^{i-1} a^{-j}$ . The loop terminates if  $R < (1 + \delta)r$ . Using the geometric series the loop terminates if  $R_0/a^i + r \cdot a/(a - 1) \leq (1 + \delta)r$ . Since  $a = 1 + 2/\delta$  and  $r \geq 1/2$ , the lemma follows.

Using this bound for the number of iterations we can analyze the running time of the sampling procedure.

**Lemma 5.** *Given a lattice basis  $B$  and a subspace  $M \subset \text{span}(\mathcal{L}(B))$ , with the parameters  $r$ ,  $a$  and  $\delta$  chosen as above, the running time of the sampling procedure [3](#) is bounded by  $((1 + 2/\delta) \cdot b \cdot N)^{\mathcal{O}(1)}$ . Here  $b$  is the size of  $\mathcal{L}(B)$  and  $M$ . Furthermore,  $N$  is the number of points chosen in the sampling procedure.*

*Proof.* The number of iterations in the while-loop is at most  $2 \log_2(1 + 2/\delta) \cdot (\log_2(1 + 2/\delta) + \log_2 R_0) \leq (1 + 2/\delta)b^{\mathcal{O}(1)}$ . In each iteration we apply the sieving procedure. Since the input size is at most  $b$  the running time of the sampling procedure is at most  $(1 + 2/\delta)N^2b^{\mathcal{O}(1)} = ((1 + 2/\delta) \cdot b \cdot N)^{\mathcal{O}(1)}$ .

Summarizing the previous results about the sampling procedure [3](#), we get

**Theorem 7.** *For every  $0 < \epsilon \leq 2$  there exists a  $\delta > 0$  such that the following holds: Given a lattice  $L = \mathcal{L}(B)$ , a subspace  $M$ , and  $r$  satisfying  $1/2 \leq r \leq (1/2) \cdot (1 + \delta)^2 \lambda_M^{(p)}(L)$ , the sampling procedure [3](#) computes a set of vectors from  $L \cap B^{(p)}(0, (1 + \epsilon)\lambda_M^{(p)}(L))$ . The running time of the sampling procedure is  $((2 + 1/\epsilon)^n \cdot b)^{\mathcal{O}(1)}$ , where  $b$  is the size of  $L$  and  $M$ .*

The proof is obviously if we choose  $\delta = (1/4)\epsilon$  and combine this with the results of Lemma [3](#) and Lemma [5](#).

Also, using Lemma [2](#) and Lemma [4](#) we get

**Lemma 6.** *If we apply the sampling procedure [3](#) with the parameters  $\delta$ ,  $a$  and  $r$  chosen as above, we remove at most*

$$z(R_0, \delta) := (\log_2 R_0 + \log_2(1 + 2/\delta))(2(1 + 2/\delta) + 1)^{n+1} \quad (2)$$

*pairs from the set  $\mathcal{Z}$ .*

### 4.3 Modification of the Sampling Procedure

We need to show that the sampling procedure [3](#) computes vectors in  $L \setminus M$ . For this we use the randomization in the algorithm. We change our point of view and consider a modified sampling procedure that behaves exactly like the sampling procedure [3](#). We are able to show that the modified sampling procedure computes with probability exponentially close to 1 a vector  $v \in L \setminus M$ . Hence, the same is true for the sampling procedure [3](#).

Let  $u \in L \setminus M$  a lattice vector with  $\|u\|_p = \lambda_M^{(p)}(L)$ . Define

$$C_1 := B^{(p)}(0, r) \cap B^{(p)}(u, r) \text{ and } C_2 := B^{(p)}(0, r) \cap B^{(p)}(-u, r).$$

If the parameter  $r$  satisfies

$$\frac{1}{2}(1 + \delta)\lambda_M^{(p)}(L) \leq r \leq \frac{1}{2}(1 + \delta)^2\lambda_M^{(p)}(L) \quad (3)$$

for a  $\delta > 0$ , the sets  $C_1$  and  $C_2$  are non-empty and disjoint. We define a bijective mapping  $\tau_u : B^{(p)}(0, r) \rightarrow B^{(p)}(0, r)$  depending on the lattice vector  $u$ .

$$\tau_u(x) = \begin{cases} x + u, & x \in C_2 \\ x - u, & x \in C_1 \\ x, & \text{otherwise} \end{cases}$$



Using the mapping  $\tau_u$  we define the modified sampling procedure (see Algorithm 4). Since the modified sampling procedure is only used for the analysis, we do not worry about its running time and the fact that it uses the unknown  $u$ . The sampling procedure 3 and the modified sampling procedure 4 return vectors in  $L \cap B^{(p)}(0, (1 + \epsilon)\lambda_M^{(p)}(L))$  distributed according to certain distributions. We call these the *output distributions* generated by the sampling procedure and the modified sampling procedure, respectively. Next, we show

**Algorithm 4 The modified sampling procedure**

*Input:* A lattice  $L = \mathcal{L}(B)$ ,  $B = \{b_1, \dots, b_n\}$ , and a subspace  $M \subseteq \text{span}(B)$

1. (a)  $R_0 \leftarrow n \cdot \max_i \|b_i\|_p$ .  
 (b) Choose  $N$  points  $x_1, \dots, x_N$  uniformly in  $B^{(p)}(0, r)$ .  
 (c) Compute  $y_i \in \mathcal{P}(B)$  with  $y_i \equiv x_i \pmod{\mathcal{L}(B)}$  for  $i = 1, \dots, N$ .  
 (d) Set  $\mathcal{Z} \leftarrow \{(x_1, y_1), \dots, (x_N, y_N)\}$  and  $R \leftarrow R_0$ .
2. While  $R > (1 + \delta)r$  do  
 (a) Apply the sieving procedure 2 to  $\{y_i | (x_i, y_i) \in \mathcal{Z}\}$  with the parameters  $a$  and  $R$ . The result is a set  $J$  and a mapping  $\eta$ .  
 (b) Remove from  $\mathcal{Z}$  all pairs  $(x_i, y_i)$  with  $i \in J$ .  
 (c) For each pair  $(x_i, y_i)$ ,  $i \in J$ , replace  $x_i$  with  $\tau_u(x_i)$  with probability  $1/2$ .  
 (d) Replace each remaining pair  $(x_i, y_i) \in \mathcal{Z}$  with  $(x_i, y_i - (y_{\eta(i)} - x_{\eta(i)}))$ .  
 (e)  $R \leftarrow \frac{R}{\alpha} + r$
3. For each pair  $(x_i, y_i) \in \mathcal{Z}$  replace  $x_i$  with  $\tau_u(x_i)$  with probability  $1/2$ .

*Output:* A shortest vector  $v \in \{y_i - x_i | (x_i, y_i) \in \mathcal{Z}\}$  with  $v \notin M$ , if such a vector exists.

**Theorem 8.** *The sampling procedure 3 and the modified sampling procedure 4 generate the same output distribution.*

*Proof.* We consider a series of modifications to the sampling procedure 3 leading to the modified sampling procedure 4. In the first modification, after choosing in step 1b the points  $x_i$  we decide for each  $x_i$  uniformly at random whether to keep  $x_i$  or to replace it with  $\tau_u(x_i)$ . Since  $\tau_u$  is bijective, this does not change the distribution on the points  $x_i$ . Hence, this modification does not change the output distribution of the sampling procedure. Next, observe that  $u \in L$  implies  $y_i \equiv x_i \equiv \tau_u(x_i) \pmod{L}$ ,  $i = 1, \dots, N$ . Hence, if we decide for each  $x_i$  whether to replace it with  $\tau_u(x_i)$  at the end of step 1 rather than in step 1b, then this does not change the output distribution.

But if, without changing the output distribution, we can choose for each  $x_i$  whether to keep it or to replace it with  $\tau_u(x_i)$  at the end of step 1, then making that decision for each  $x_i$  prior to the first time it is used in step 2 will also not change the output distribution. Furthermore, for each point  $x_i$  not used at all in step 2 we can choose whether to keep it or replace it with  $\tau_u(x_i)$  at the end of step 2. But this is exactly the modification leading from the sampling procedure 3 to the modified sampling procedure 4.

For the further analysis only pairs  $(x_i, y_i)$  with  $x_i \in C_1 \cup C_2$  are of interest because only for them the mapping  $\tau_u$  is not the identity. In the following, three lemmata we will show that with high probability at the end of the sampling procedure [3](#) or the modified sampling procedure [4](#) the set  $\mathcal{Z}$  contains at least  $2^n$  pairs with this property. All lemmata are stated without proof. First, we need the probability, that a point  $x$ , which is chosen uniformly in  $B^{(p)}(0, r)$ , is contained in  $C_1 \cup C_2$ .

**Lemma 7.** *Let  $u \in \mathbb{R}^n$  be a vector with  $\|u\|_p = \rho$  and  $\zeta > 0$ . Define  $C = B^{(p)}(0, (1/2)(1 + \zeta)\rho) \cap B^{(p)}(u, (1/2)(1 + \zeta)\rho)$ . Then*

$$\frac{\text{vol}(C)}{\text{vol}(B^{(p)}(0, \frac{1}{2}(1 + \zeta)\rho))} \geq 2^{-n} \left( \frac{\zeta}{1 + \zeta} \right)^n.$$

Next, we are interested in the number of points  $x_i$ , which are contained in  $C_1 \cup C_2$ , if we choose  $N$  points uniformly at random in  $B^{(p)}(0, r)$ .

**Lemma 8.** *Let  $N \in \mathbb{N}$ . By  $q$  denote the probability that a random point in  $B^{(p)}(0, r)$  is contained in  $C_1 \cup C_2$ . If  $N$  points  $x_1, \dots, x_N$  are chosen uniformly at random in  $B^{(p)}(0, r)$ , then with probability larger than  $1 - 4/(N \cdot q)$ , there are at least  $(q \cdot N)/2$  points  $x_i \in \{x_1, \dots, x_N\}$  with the property  $x_i \in C_1 \cup C_2$ .*

From the Lemmata [7](#) and [8](#) combined with Lemma [6](#) we obtain

**Lemma 9.** *Let  $L$  be a lattice and  $M \subset \text{span}(L)$  be a subspace. Furthermore, assume that in the first step of the sampling procedure [3](#) or of the modified sampling procedure [4](#) the number of points chosen is  $N = ((1 + \delta)/\delta)^n 2^{n+1} (2^n + z(R_0, \delta))$ , where  $z(R_0, \delta)$  is defined as in [2](#). Then at the end of step [2](#) of the sampling procedure [3](#) or the modified sampling procedure [4](#) the set  $\mathcal{Z}$  contains with probability exponentially close to 1 at least  $2^n$  pairs  $(x, y)$  with the property  $x \in C_1 \cup C_2$ .*

**Theorem 9.** *For every  $0 < \epsilon \leq 2$  there exists a  $\delta > 0$  such that the following holds: Given a lattice  $L = \mathcal{L}(B)$ , a subspace  $M$  of  $\text{span}(L)$ , for which  $2 \leq \lambda_M^{(p)}(L)$  and  $r$  satisfying [3](#), the modified sampling procedure [4](#) computes with probability exponentially close to 1 a vector  $v \in L \setminus M$ .*

*Proof.* We apply the modified sampling procedure with the same parameter as in Theorem [7](#), i. e.  $\delta = (1/4)\epsilon$ . Since  $2 \leq \lambda_M^{(p)}(L)$ , we have  $r \geq 1/2$ . By assumption  $u \in L \setminus M$ . If  $y - x \in M$ , then  $y - \tau_u(x) = y - x \pm u \in L \setminus M$ . The modified sampling procedure returns a vector  $v \in L \setminus M$ , if at the end of step [2](#) there exists a pair  $(x, y) \in \mathcal{Z}$  with  $x \in C_1 \cup C_2$  and one of the following conditions holds:  $y - x \in M$  and in step [3](#) we replace  $x$  with  $\tau_u(x)$  or  $y - x \in L \setminus M$  and in step [3](#) we do not replace  $x$  with  $\tau_u(x)$ . In step [3](#) of the modified sampling procedure we decide for each pair  $(x, y) \in \mathcal{Z}$  uniformly if we replace it or not. Using Lemma [9](#) the set  $\mathcal{Z}$  contains with probability exponentially close to 1 at least  $2^n$  pairs  $(x, y)$  with the property  $x \in C_1 \cup C_2$ . Therefore the probability, that the modified sampling procedure does not return a vector  $v \in L \setminus M$ , is bounded by  $2^{-2^n}$ .

By Theorem 8 the sampling procedure and the modified sampling procedure generate the same output distribution. Also, we have shown that we can restrict ourselves to instances of SAP with  $2 \leq \lambda_M^{(p)}(L) < 3$  (Lemma 11). Hence, we get

**Theorem 10.** *There exists a randomized algorithm that for all  $\ell_p$  norms,  $1 \leq p \leq \infty$ , solves SAP with approximation factor  $1 + \epsilon$ ,  $0 < \epsilon \leq 2$  arbitrary, with probability exponentially close to 1. The running time of the algorithm is  $((2 + 1/\epsilon)^n \cdot b)^{\mathcal{O}(1)}$ , where  $b$  is the size of the input lattice and the subspace.*

*Proof.* Let  $\delta = (1/4)\epsilon$ . Using Lemma 11 we can assume:  $2/3 < 2/\lambda_M^{(p)}(L) \leq 1$ . Let  $\kappa_0 = \log_{1+\delta}(2/3)$  and  $\kappa_1 = 0$ . Set  $l := \lceil \log_{1+\delta} 2/\lambda_M^{(p)}(L) \rceil$ , then  $\kappa_0 \leq l \leq \kappa_1$  and  $r := (1+\delta)^{2-l}$  satisfies the Equation (3), i.e.,  $(1/2)(1+\delta)\lambda_M^{(p)}(L) \leq r \leq (1/2)(1+\delta)^2\lambda_M^{(p)}(L)$ . We apply the sampling procedure for each value  $r = (1+\delta)^{2-l'}$  with  $\kappa_0 \leq l' \leq \kappa_1$ . Let  $v_{l'} \in L \setminus M$  be the lattice point discovered by the sampling procedure started with  $r = (1+\delta)^{2-l'}$ , if any lattice point is discovered. The output will be the smallest  $v_{l'} \in L \setminus M$ . As we have seen, for the unique  $l' = l$  such that  $r = (1+\delta)^{2-l'}$  satisfies the Equation (3) the sampling procedure will find a  $(1+\epsilon)$ -approximation for SAP with probability exponentially close to 1.

We apply the sampling procedure  $\lfloor \log_{1+\delta}(2/3) \rfloor$  times. By our choice of  $\delta = (1/4)\epsilon$  the running time is  $\lfloor \log_{1+\delta} 2/3 \rfloor ((2 + 1/\epsilon)^n \cdot b)^{\mathcal{O}(1)} = ((2 + 1/\epsilon)^n \cdot b)^{\mathcal{O}(1)}$ .

Combining this with the results from Section 3 we obtain:

**Theorem 11.** *There exist randomized algorithms that for all  $\ell_p$  norms,  $1 \leq p \leq \infty$ , approximate SVP, SMP, SIVP, and CVP with probability exponentially close to 1. In case of SVP, SMP, and SIVP the approximation factor is  $1 + \epsilon$  for any  $0 < \epsilon \leq 2$ . For CVP the approximation factor is  $1 + \epsilon$  for any  $0 < \epsilon < 1/2$ . The running time of the algorithms is  $((2 + 1/\epsilon)^n \cdot b)^{\mathcal{O}(1)}$ , where  $b$  is the size of the input lattice and the subspace.*

## 5 Using the Sampling Procedure for Optimal Solutions

**Theorem 12.** *Let  $L = \mathcal{L}(B)$  be a lattice and  $M$  be a subspace of  $\text{span}(L)$ , both of size  $b$ . Assume that there exist absolute constants  $c, \epsilon$  such that the number of  $v \in L \setminus M$  satisfying  $\|v\|_p \leq (1+\epsilon)\lambda_M^{(p)}(L)$  is bounded by  $2^{cn}$ . Then there exists an algorithm that solves SAP with probability exponentially close to 1. The running time is  $(2^n \cdot b)^{\mathcal{O}(1)}$ .*

*Proof.* To turn the  $(1+\epsilon)$ -sampling procedure into an exact algorithm, we use the sampling procedure 3 with the parameters  $\delta = (1/4)\epsilon$  and  $N = ((1+\delta)/\delta)^n 2^{n+1} (5 \cdot 2^{(c+1)n} + z(R_0, \delta))$ , where  $z(R_0, \delta)$  is defined in (2). We only modify the output: We consider the two sets

$$O_1 := \{(y_i - x_i) - (y_j - x_j) \mid (x_i, y_i), (x_j, y_j) \in \mathcal{Z}\} \text{ and } O_2 := \{y_i - x_i \mid (x_i, y_i) \in \mathcal{Z}\}.$$

The output is a shortest lattice vector  $v \in O_1 \cup O_2$  with  $v \in L \setminus M$ . The analysis and the running time of this sampling procedure are the same as in Section 4.

Obviously, we can modify the sampling procedure in the same way as in Theorem 8 by using the mapping  $\tau_u$  with respect to a shortest vector  $u \in L \setminus M$ . We obtain a modified sampling procedure similar to procedure 4 which generates the same output distribution as the original sampling procedure. Hence, we only need to analyze the success probability of the modified sampling procedure. We show that the modified sampling procedure computes with probability exponentially close to 1 the lattice vector  $u$ . In the following, consider the set  $\mathcal{Z}$  after step 2 and before step 3 of the modified sampling procedure. We define the multiset  $F := \{(x_i, y_i) \in \mathcal{Z} | x_i \in C_1\} \subseteq \mathcal{Z}$  and for  $v \in L$  we set  $F_v := \{(x_i, y_i) \in F | y_i - x_i = v\}$ . As in Lemma 9, we can show, that  $F$  contains with probability exponentially close to 1 at least  $5 \cdot 2^{(c+1)n}$  pairs. Next, we consider two cases: 1) There exists an  $v \in L$  with  $|F_v| \geq 2^n$  and 2)  $|F_v| < 2^n$  for all  $v \in L$ .

In the first case, in step 3 we decide for each pair  $(x, y) \in F_v$  uniformly whether we replace  $x$  with  $\tau_u(x)$  or not. If there exist  $(x_i, y_i), (x_j, y_j) \in F_v$  such that in step 3 the mapping  $\tau$  is applied to  $x_i$  but not to  $x_j$  then  $u \in O_1$ . This event happens with probability  $1 - 2 \cdot 2^{-2^n}$ .

In the second case, we show that with probability exponentially close to 1 the vector  $u$  is contained in the set  $O_2$ . We do this by showing that after step 3 of the modified sampling procedure all vectors  $v \in L \setminus M$  satisfying  $\|v\|_p \leq (1 + \epsilon)\lambda_M^{(p)}(L)$  are contained in  $O_2$ . In the following, we consider  $\mathcal{F} := \{v \in L | \exists (x, y) \in F \text{ with } v = y - x\}$ . Since  $|F| > 5 \cdot 2^{(c+1)n}$  and  $|F_v| < 2^n$  for all  $v \in L$ , we obtain  $|\mathcal{F}| \geq 5 \cdot 2^{cn}$ . Let  $\mathcal{F}_1 := \mathcal{F} \cap M$ . By assumption  $|\mathcal{F} \setminus \mathcal{F}_1| \leq 2^{cn}$  and therefore  $|\mathcal{F}_1| = |\mathcal{F}| - |\mathcal{F} \setminus \mathcal{F}_1| \geq 2^{2cn}$ . For all  $v = y - x \in \mathcal{F}_1$  we have  $y - \tau_u(x) \in L \setminus M$ . Analogously to Lemma 8, we can show that with probability exponentially close to 1, for at least  $2^{2cn}$  elements  $v = y - x$  in  $\mathcal{F}_1$  we replace in step 3 the element  $x$  by  $\tau_u(x)$ . Hence, with probability exponentially close to 1 we get  $2^{2cn}$  elements  $y - x - u \in L \setminus M$ . All these elements have length at most  $(1 + \epsilon)\lambda_M^{(p)}(L)$ . Combining this with  $|(L \setminus M) \cap B^{(p)}(0, (1 + \epsilon)\lambda_M^{(p)}(L))| < 2^{2cn}$  we see that in this case the set  $O_2$  contains all vectors  $v \in L \setminus M$  of length at most  $(1 + \epsilon)\lambda_M^{(p)}(L)$ .

To use the exact sampling procedure to solve SVP, CVP, SMP and SIVP we need the following lemma, whose proof is almost identical to the proof of Lemma 2.

**Lemma 10.** *Let  $L$  be a lattice and  $R > 0$ . Then*

$$|B^{(p)}(0, R) \cap L| < \left( (2R + \lambda_1^{(p)}(L)) / \lambda_1^{(p)}(L) \right)^n.$$

In case of SVP we use Lemma 10 with  $R = (1 + \epsilon)\lambda_1^{(p)}(L)$  and get  $|B^{(p)}(0, (1 + \epsilon)\lambda_1^{(p)}(L)) \cap L| \leq (3 + 2\epsilon)^n = 2^{cn}$  for a  $c \in \mathbb{N}$ . Therefore, the assumptions of Theorem 12 are satisfied in case of SVP and we obtain the following.

**Theorem 13.** *Let  $L \subset \mathbb{Q}^n$  be a lattice of size  $b$ . A shortest non-zero vector in  $L$  can be computed with probability exponentially close to 1. The running time is  $(2^n \cdot b)^{O(1)}$ .*

Using Lemma [10](#), for SMP and CVP we can only show that the number of almost optimal solutions to SMP or CVP is single exponential in the rank of  $L$  if the  $n$ -th successive minimum  $\lambda_n^{(p)}(L)$  or the distance  $D_t$  of target vector  $t$  to lattice  $L$  are bounded by  $c\lambda_1^{(p)}(L)$  for some constant  $c$ . Hence, we get

**Theorem 14.** *Let  $L \subset \mathbb{Q}^n$  be a lattice of size  $b$ . Assume that the  $n$ -th successive minimum  $\lambda_n^{(p)}$  is bounded by  $c\lambda_1^{(p)}$  for some constant  $c \in \mathbb{N}$ . Then the successive minima of  $L$  can be computed with probability exponentially close to 1. The running time is  $(2^n \cdot b)^{\mathcal{O}(1)}$ .*

**Theorem 15.** *Let  $c > 0$  be some constant. Assume lattice  $L \in \mathbb{Q}^n$  and target vector  $t \in \text{span}(L)$  are of size  $b$ . Assume furthermore, that  $D_t \leq c\lambda_1^{(p)}(L)$ . Then a vector  $v \in L$  satisfying  $\|t - v\|_p = D_t$  can be computed with probability exponentially close to 1. The running time is  $(2^n \cdot b)^{\mathcal{O}(1)}$ .*

*Acknowledgment.* We thank O. Regev for several stimulating discussions that greatly benefited the paper. Moreover, his lecture notes on the Ajtai, Kumar, Sivakumar algorithm for SVP [Reg04](#) were the starting point for our research.

## References

- [Ajt98] Ajtai, M.: The shortest vector problem in  $l_2$  is NP-hard for randomized reductions. In: Proceedings of the 30th ACM Symposium on Theory of Computing, pp. 10–19. ACM Press, New York (1998)
- [AKS01] Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the 33th ACM Symposium on Theory of Computing, pp. 601–610. ACM Press, New York (2001)
- [AKS02] Ajtai, M., Kumar, R., Sivakumar, D.: Sampling short lattice vectors and the closest lattice vector problem. In: Proceedings of the 17th IEEE Annual Conference on Computational Complexity - CCC, pp. 53–57. IEEE Computer Society Press, Los Alamitos (2002)
- [Bab86] Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica* 6(1), 1–13 (1986)
- [Blö00] Blömer, J.: Closest vectors, successive minima, and dual HKZ-bases of lattices. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 248–259. Springer, Heidelberg (2000)
- [BS99] Blömer, J., Seifert, J.-P.: The complexity of computing short linearly independent vectors and sort bases in a lattice. In: Proceedings of the 21th Symposium on Theory of Computing, pp. 711–720 (1999)
- [DFK91] Dyer, M., Frieze, A., Kannan, R.: A random polynomial time algorithm for approximating the volume of convex bodies. *Journal of the ACM* 38(1), 1–17 (1991)
- [DKRS03] Dinur, I., Kindler, G., Raz, R., Safra, S.: Approximating CVP to within almost-polynomial factors in NP-hard. *Combinatorica* 23(2), 205–243 (2003)
- [Kan87a] Kannan, R.: Algorithmic geometry of numbers. *Annual Reviews in Computer Science* 2, 231–267 (1987)

- [Kan87b] Kannan, R.: Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research* 12(3), 415–440 (1987)
- [Kho05] Khot, S.: Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM (JACM)* 52(5), 789–808 (2005)
- [LLL82] Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. *Mathematische Annalen* 261, 515–534 (1982)
- [MG02] Micciancio, D., Goldwasser, S.: *Complexity of Lattice Problems - A Cryptographic Perspective*. Kluwer Academic Publishers, Dordrecht (2002)
- [Mic00] Micciancio, D.: The shortest vector in a lattice is hard to approximate to within some constant. *SIAM Journal on Computing* 30(6), 2008–2035 (2000)
- [Reg04] Regev, O.: *Lecture note on lattices in computer science, lecture 8:  $2^{O(n)}$ -time algorithm for SVP* (2004)
- [Sch87] Schnorr, C.-P.: A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science* 53, 201–224 (1987)
- [Sch94] Schnorr, C.-P.: Block reduced lattice bases and successive minima. *Combinatorics, Probability & Computing* 3, 507–522 (1994)

# Low Distortion Spanners

Seth Pettie

The University of Michigan

**Abstract.** A *spanner* of an undirected unweighted graph is a subgraph that approximates the distance metric of the original graph with some specified accuracy. Specifically, we say  $H \subseteq G$  is an  $f$ -spanner of  $G$  if any two vertices  $u, v$  at distance  $d$  in  $G$  are at distance at most  $f(d)$  in  $H$ . There is clearly some tradeoff between the sparsity of  $H$  and the distortion function  $f$ , though the nature of this tradeoff is still poorly understood.

In this paper we present a simple, modular framework for constructing sparse spanners that is based on interchangeable components called *connection schemes*. By assembling connection schemes in different ways we can recreate the additive 2- and 6-spanners of Aingworth et al. and Baswana et al. and improve on the  $(1 + \epsilon, \beta)$ -spanners of Elkin and Peleg, the sublinear additive spanners of Thorup and Zwick, and the (non constant) additive spanners of Baswana et al. Our constructions rival the simplicity of all comparable algorithms and provide substantially better spanners, in some cases reducing the density *doubly* exponentially.

## 1 Introduction

An  $f$ -spanner of an undirected, unweighted graph  $G$  is a subgraph  $H$  such that

$$\delta_H(u, v) \leq f(\delta_G(u, v))$$

holds for every pair of vertices  $u, v$ , where  $\delta_H$  is the distance metric w.r.t.  $H$ . The premier open problem in this area is to understand the necessary tradeoffs between the sparsity of  $H$  and the distortion function  $f$ . The problem of finding a sparse spanner is one in the wider area of *metric embeddings*, where *distortion* is almost universally defined to be multiplicative, of the form  $f(d) = t \cdot d$  for some  $t \geq 1$ . Spanners, however, can possess substantially stronger properties. The recent work of Elkin and Peleg [11] and Thorup and Zwick [22] shows that the multiplicative distortion  $f(d)/d$  can tend toward 1 as  $d$  increases; in this situation the nature of the tradeoff is between the sparsity of the spanner and the *rate* of convergence. It is unknown whether this type of tradeoff is the best possible or whether there exist arbitrarily sparse *additive* spanners, where  $f(d) = d + O(1)$  and the tradeoff is between sparsity and the constant hidden in the  $O(1)$  term.

*Applications.* The original application of spanners was in the efficient simulation of synchronized protocols in unsynchronized networks [3, 17]. Thereafter spanners

were used in the design of low-stretch routing schemes using small routing tables (see [9,19,21] and the references therein), computing almost shortest paths in distributed networks [12], and in approximation algorithms for geometric spaces [15]. A recent application of spanners is in the design of approximate distance oracles and labeling schemes for arbitrary metrics; see [23,4] for further references. Tree spanners have found a number of uses in recent years, such as solving diagonally dominant linear systems [20] and various approximation algorithms [13]. (Tree spanners cannot have any non-trivial distortion in the worst case so weaker notions are used, such as average distortion and expected distortion over a distribution of spanning trees.) In all the applications cited above the quality of the solution is directly related to the quality of the underlying spanners.

*Sparseness-Distortion Tradeoffs.* It was observed early on [16,2] that a spanner has multiplicative distortion  $t$  if and only if  $f(1) = t$ , that is, if the distance between adjacent vertices in  $G$  is at most  $t$  in the spanner  $H$ . Althöfer et al. [2] proved that the sparsest multiplicative  $t$ -spanner has precisely  $m_{t+2}(n)$  edges, where  $m_g(n)$  is the maximum number of edges in a graph with  $n$  vertices and girth at least  $g$ .<sup>1</sup> The upper bound follows from a trivial greedy algorithm (similar to Kruskal's minimum spanning tree algorithm) and the lower bound is also simple. In any graph with girth  $t + 2$ , removing any edge shifts the distance of its endpoints from 1 to at least  $t+1$ . Thus, the only multiplicative  $t$ -spanner is the graph itself. It is easy to show that  $m_{2k+1}(n)$  and  $m_{2k+2}(n)$  are  $O(n^{1+1/k})$  and it has been conjectured by Erdős and others (see [24,23]) that this bound is asymptotically tight. However, it has only been proved for  $k = 1, 2, 3$ , and 5; see [24,23] for a longer discussion on the girth conjecture. The tradeoff between sparseness and  $f(1)$  is fully understood inasmuch as it amounts to proving the girth conjecture. The only other situation that is understood to a similar degree is the threshold  $D$  beyond which  $f$  is isometric, i.e., where  $f(d) = d$ , for all  $d \geq D$ . Bollobas et al. [6] showed that these so called distance preservers have  $\Theta(n^2/D)$  edges. The only known lower bound for an intermediate distance was given recently by Woodruff [25], who showed that  $f(k) < 3k$  holds only if the spanner has  $\Omega(k^{-1}n^{1+1/k})$  edges.<sup>2</sup> For this size spanner the best upper bound on  $f(k)$  is  $k^{\log 3}$ , which we show in Section 3.

It is perfectly consistent with the girth conjecture and Woodruff's lower bound [25] that there are spanners with size  $O(n^{1+1/k})$  and constant additive distortion  $f(d) = d + 2k - 2$ , though little progress has been made in proving or disproving their existence. Aingworth et al. [1] (see also [10,11,22]) showed that there are additive 2-spanners with size  $O(n^{3/2})$ , which is optimal, and Baswana et al. [5] gave an additive 6-spanner with size  $O(n^{4/3})$ . Below the  $O(n^{4/3})$  threshold the best known tradeoff is quite weak; it is shown in [5] that there is an  $O(n^{1+\epsilon})$ -sized spanner with  $f(d) = d + O(n^{1-3\epsilon})$ , for any  $\epsilon \in (0, 1/3)$ .

One nice property of additive spanners is that  $f(d)/d$  quickly tends toward 1 as  $d$  increases. Elkin and Peleg [11] and Thorup and Zwick [22] have shown

<sup>1</sup> Girth is the length of the shortest cycle.

<sup>2</sup> Woodruff expressed this result as a lower bound on the *additive* distortion.



that this property can be achieved without directly addressing the problem of guaranteeing a constant additive distortion. Elkin and Peleg [11] define an  $(\alpha, \beta)$ -spanner to be one with distortion  $f(d) = \alpha d + \beta$ . They show the existence of  $(1 + \epsilon, \beta)$ -spanners with size  $O(\beta n^{1+1/k})$ , where  $\beta$  is roughly  $(\epsilon^{-1} \log k)^{\log k}$ . Thorup and Zwick [22] gave a remarkably simple spanner construction with similar but incomparable properties. They showed that there is a  $O(kn^{1+1/k})$ -size  $(1 + \epsilon, O(\lceil 1 + \frac{2}{\epsilon} \rceil^{k-2}))$ -spanner, which holds for *all*  $\epsilon$  simultaneously. When  $\epsilon^{-1}$  is chosen to be  $\Theta(d^{1/(k-1)})$  the distortion function is  $f(d) = d + O(d^{1-1/(k-1)} + 2^k)$ . Notice that the  $\beta$  of the Thorup-Zwick spanner is exponentially larger than that of Elkin and Peleg.

*Our Results.* In this paper we present a simple, modular framework for constructing low distortion spanners that generalizes much of the recent work on additive and  $(\alpha, \beta)$ -spanners. In our framework a spanner is expressed as a list of *connection schemes*, which are essentially interchangeable components that can be combined in various ways. This framework simplifies the construction of spanners and greatly simplifies their analysis. Once the list of connection schemes is fixed the size and distortion of the spanner follow from some straightforward linear recurrences. In our framework it is possible to succinctly express the additive 2-spanners of [11, 22] and the additive 2- and 6-spanners of [5], as well as the additive 4-spanner suggested in [7]. By properly combining connection schemes we can simultaneously improve the sparseness and distortion of both the Elkin-Peleg and Thorup-Zwick spanners.

One nice feature of our framework is that it is possible to obtain *linear* size spanners with relatively good distortion. Previous to this work the only linear size spanners [2, 14] had  $O(\log n)$  multiplicative distortion. (The Elkin-Peleg spanners always have  $\Omega(n(\epsilon^{-1} \log \log n)^{\log \log n})$  edges. The size of the Thorup-Zwick spanners is  $\Omega(n \log n)$ , though at this sparsity the guaranteed distortion is quite weak.) We can construct an  $O(n)$ -size  $(5 + \epsilon, \beta)$ -spanner, where  $\epsilon > 0$  is constant and  $\beta = \text{polylog}(n)$ , as well as an additive  $\tilde{O}(n^{9/16})$ -spanner. Under relatively mild assumptions we can actually push the density *and* multiplicative distortion arbitrarily close to 1. For graphs with *quadratic expansion* there are  $(1 + \epsilon, \beta)$ -spanners with  $(1 + \epsilon)n$  edges, for any  $\epsilon > 0$ . By quadratic expansion we mean that the number of vertices within distance  $d$  of any vertex is  $\Omega(d^2)$ .

## 2 Notation and Overview

Throughout the paper  $G = (V, E)$  denotes the input graph. We denote by  $\delta_H(u, v)$  and  $P_H(u, v)$  the distance from  $u$  to  $v$  in  $H$  and the associated shortest path, respectively. In general there are many shortest paths between two vertices. We insist that if  $x, y \in P_H(u, v)$  then  $P_H(x, y) \subseteq P_H(u, v)$ . Whenever  $H$  is omitted it is assumed to be  $G$ . Our spanner constructions all refer to vertex sets  $V_0, V_1, \dots, V_o$ , where  $V_0 = V$  and  $V_i$  is derived by sampling  $V_{i-1}$  with probability  $q_i/q_{i-1}$ , where  $1 = q_0 > q_1 > \dots > q_o$ . Thus, the expected size of  $V_i$  is  $nq_i$ . Let  $p_i(v)$  be the closest vertex in  $V_i$  to  $v$ , breaking ties arbitrarily, and let  $\text{rad}_i(v) = \delta(v, p_i(v))$ . If  $i = o + 1$  then  $p_{o+1}(v)$  is non-existent

| Connection Scheme | Connected Pairs   | Distortion          | Expected Size              |
|-------------------|---|---------------------|----------------------------|
| <b>A</b>          | $V_i \times \bar{\mathcal{B}}_i(\cdot)$                                     | exact               | $nq_i/q_{i+1}$             |
| <b>B</b>          | $V_i \times \bar{\mathcal{B}}_i^-(\cdot)$                                   | $d + 2(\log d + 1)$ | $n\sqrt{q_i/q_{i+1}}$      |
|                   | $V_i \times \bar{\mathcal{B}}_i^{1/2}(\cdot) \cap V_i$                      | $d + 2$             | $n\sqrt{q_i/q_{i+1}}$      |
| <b>C</b>          | $V_o \times V_o$  | $d + 2$             | $n\sqrt{nq_o}$             |
|                   | $V_i \times \bar{\mathcal{B}}_i^{1/3}(\cdot) \cap V_i$                      | exact               | $n + nq_i^2/q_{i+1}^{3/2}$ |
| <b>D</b> ( $r$ )  | $V_o \times V_o$  | exact               | $n + n^{5/2}q_o^2$         |
| <b>D</b> ( $r$ )  | $V_i \times \bar{\mathcal{B}}_i(\cdot) \cap \text{Ball}(\cdot, r) \cap V_i$ | exact               | $nrq_i^2/q_{i+1}$          |
| <b>x</b>          | $V_i \times p_{i+1}(\cdot)$   | exact               | $n$                        |

**Fig. 1.** The connection schemes. Here  $0 \leq i \leq o$ . Schemes **B** and **C** have slightly stronger guarantees at  $i = o$ . The notation  $V_i \times \bar{\mathcal{B}}_i(\cdot)$  is short for  $\{(u, v) : u \in V_i, v \in \bar{\mathcal{B}}_i(u)\}$ .

and  $\text{rad}_{o+1}(v) = \infty$  by definition. Let  $\text{Ball}(v, r) = \{u : \delta(v, u) < r\}$ . We define  $\mathcal{B}_i^\epsilon(v) = \text{Ball}(v, \epsilon \cdot \text{rad}_{i+1}(v))$  and  $\mathcal{B}_i^-(v) = \text{Ball}(v, \text{rad}_{i+1}(v) - 1)$ , where  $\epsilon$  is taken to be 1 if omitted. Let  $\bar{\mathcal{B}}_i^x(v) = \mathcal{B}_i^x(v) \cup \{p_{i+1}(v)\}$ , where  $x$  is ‘-’ or some  $\epsilon$ .

In Section 4 we describe five connection schemes called **A**, **B**, **C**, **D**, and **x**. In our framework a spanner can be expressed by choosing an *order*  $o$  and a list of the connection schemes employed at each level. For instance, in our compact notation the spanner **ABB** employs scheme **A** at level zero and scheme **B** at levels 1 and 2, where in this case  $o = 2$ . When a connection scheme is employed at level  $i$  it returns a subgraph that connects each  $v \in V_i$  to some subset of the vertices in  $\bar{\mathcal{B}}_i(v)$ ; the particulars depend on the scheme used. The overall properties of the spanner are determined by the sequence of connection schemes and, in general, a larger order  $o$  leads to a sparser spanner with higher distortion. Figure 1 lists the specifications for the different schemes and Figure 2 lists some of the interesting spanners that can be generated from  $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{x}\}^*$ .

The connection schemes **A**, **B**, **C**, **D** and **x** all produce subgraphs that connect certain pairs of vertices by shortest or almost shortest paths. The three features of a connection scheme we care about are the pairs of vertices to be connected, the guaranteed distortion, and the expected size of the subgraph as a function of the sampling probabilities. The properties of each of the connection schemes are given in Figure 1. (Notice that some of the connection schemes have slightly stronger properties when used at the highest level  $o$ .) Let us decipher a few of the lines in Figure 1. When **A** is used at level  $i$  it returns a subgraph  $H_i$  such that for  $v \in V_i$  and  $u \in \bar{\mathcal{B}}_i(v)$ ,  $\delta_{H_i}(v, u) = \delta(v, u)$ , and furthermore, the expected size of  $H_i$  is on the order of  $nq_i/q_{i+1}$ . (The notation  $V_i \times \bar{\mathcal{B}}_i(\cdot)$  is short for the set of pairs  $\{(v, u) : v \in V_i, u \in \bar{\mathcal{B}}_i(v)\}$ .) Like **A**, schemes **C** and **D** have no distortion but connect fewer pairs of vertices. For  $v \in V_i$ , scheme **C** only connects the pair  $(v, u)$  if  $u$  is in *both*  $V_i$  and  $\bar{\mathcal{B}}_i^{1/3}(v)$ . Scheme **D**( $r$ ) requires  $u$  to be in  $V_i$ ,

| Encoding  | Distortion $f(d)$        | Size   | Notes                  |
|---|--------------------------|--|------------------------|
| <b>A</b> <sup>2</sup> or <b>B</b>                           | $d + 2$                  | $O(n^{3/2})$                                   | [1,10,11,5,22]         |
| <b>AC</b>   | $d + 4$                  | $O(n^{3/2}), \Omega(n^{4/3})$                  | [7]                    |
| <b>AB</b>   | $d + 6$                  | $O(n^{4/3})$                                   | [5]                    |
| <b>A</b> <sup><math>o+1</math></sup>                        | $d + O(d^{1-1/o} + 3^o)$ | $O(on^{1+1/o})$                                | [22]                   |
| not appl.   | $d + O(d^{1-1/o} + 2^o)$ | $O(on^{1+1/o})$                                | [22]                   |
| <b>AB</b> <sup>2</sup>                                      | $d + O(\sqrt{d})$        | $O(n^{6/5})$                                   | <b>new</b>             |
| <b>AB</b> <sup>2</sup> <b>C</b>                             | $d + O(d^{2/3})$         | $O(n^{25/22})$                                 | <b>new</b>             |
| <b>AB</b> <sup>2</sup> <b>C</b> <sup><math>o-2</math></sup> | $d + O(od^{1-1/o})$      | $O(on^{1+\frac{(3/4)^{o-2}}{7-2(3/4)^{o-2}}})$ | <b>new</b> , $d > o^o$ |

| Very Sparse Spanners   |   |                                |  |
|--|---|--------------------------------|--|
| not appl.  | $O(d \log n)$                                   | $O(n)$                         | [2,14]                                     |
| <b>AD</b> <sup><math>\log \log n</math></sup>                | $(5 + \epsilon)d$                               | $O(n)$                         | <b>new</b> , $d > \epsilon^{-\log \log n}$ |
| <b>xCC</b>   | $d + \tilde{O}(n^{9/16})$                       | $O(n)$                         | <b>new</b>                                 |
| <b>xD</b> <sup><math>\log \log n</math></sup>                | $(1 + \epsilon)d + \beta''$                     | $(1 + \epsilon)n$              | <b>new</b> , see below                     |
| not appl.  | $(1 + \epsilon)d + \beta$                       | $O(n\beta)$                    | [11]                                       |
| <b>AC</b> <sup><math>O(\log \log \epsilon^{-1})</math></sup> | $(1 + \epsilon)d + \beta'$                      | $O(n \log \log \epsilon^{-1})$ | <b>new</b> , $\epsilon < 1/\log \log n$    |
| <b>D</b> <sup><math>\log \log n</math></sup>                 | $(1 + \epsilon)d + \beta'$                      | $O(n \log \log \epsilon^{-1})$ | <b>new</b> , $\epsilon < 1/\log \log n$    |
| <b>AC</b> <sup><math>O(\log \log n)</math></sup>             | $d + \tilde{O}(d^{1-\frac{1}{O(\log \log n)}})$ | $O(n \log \log n)$             | <b>new</b> , $d > \text{polylog}(n)$       |

**Fig. 2.** Some of the spanners generated by  $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}(\cdot), \mathbf{x}\}^*$ . Here  $\beta, \beta'$ , and  $\beta''$  are all  $O(\epsilon^{-1} \log \log n)^{\log \log n}$ .

$\bar{\mathcal{B}}_i(v)$ , and  $\text{Ball}(v, r)$ , where  $r$  is a given parameter that influences the size of the subgraph. Scheme **B** guarantees two grades of distortion. If  $u$  is in both  $\bar{\mathcal{B}}_i^{1/2}(v)$  and  $V_i$  the additive distortion is 2 and if  $u$  is in  $\bar{\mathcal{B}}_i^-(v)$  the additive distortion is  $2(\log d + 1)$ , where  $d = \delta(v, u)$ . Scheme **x** simply connects every  $v \in V_i$  to the nearest vertex  $p_{i+1}(v) \in V_{i+1}$ .

In Section 3 we show how connection schemes can be composed in various ways to yield spanners with different sparseness-distortion tradeoffs. The construction and analysis of these spanners is inspired by the *distance emulators* of Thorup and Zwick [22]. In Section 4 we present the algorithms behind schemes **A** and **C**. See [18] for a description of the other schemes.

### 3 Modular Spanner Construction

In our framework a spanner is expressed as a finite sequence of connection schemes. For instance, the spanner **ABB** consists of the edge sets  $H_0, H_1$ , and  $H_2$ , where  $H_0$  is the subgraph returned by the connection scheme **A** applied to the zeroth level, and  $H_1$  and  $H_2$  are the subgraphs returned by applying **B** to levels 1 and 2. The size of the spanner depends solely on the sampling probabilities and there is typically one optimal choice of probabilities. For instance, for

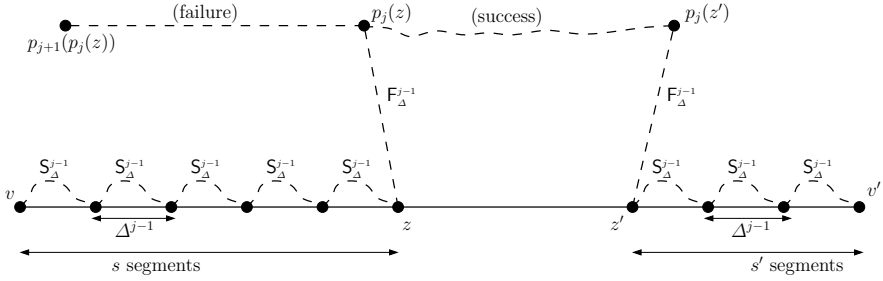
the spanner **ABB** the expected size is asymptotically  $n/q_1 + n\sqrt{q_1/q_2} + n\sqrt{nq_2}$ , which is optimized at  $q_1 = n^{-1/5}$ ,  $q_2 = n^{-3/5}$ . The distortion of the spanner is analyzed by solving some linear recurrences. The derivation of these recurrences is sketched below and formally proved in Lemma 1. Lemma 2 solves these recurrences for the class of spanners that use schemes **A**, **B**, and **C**. These three schemes are sufficient to prove our strongest sparseness-distortion trade-offs (Theorem 1) but they are ultimately incapable of generating spanners with  $o(n \log \log n)$  edges. See the full version [18] for constructions of sparser spanners.

Suppose we have a spanner  $H$  defined by a finite string  $\tau \in \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}^{o+1}$ . Let  $\tau(j)$  be the  $j$ th character of  $\tau$ , for some  $0 \leq j \leq o$ , and let  $H_j \subseteq H$  be the subgraph returned by the connection scheme  $\tau(j)$  at level  $j$ . Let  $v$  and  $v'$  be two vertices at distance at most  $\Delta^j$  in the original graph, where  $\Delta \geq 2$  is an integer. To get from  $v$  to  $v'$  in  $H$  we divide up  $P(v, v')$  into segments of length  $\Delta^{j-1}$ ; see Figure 3. Let  $v_\ell$  be the first vertex in the  $\ell$ th segment. Using only the subgraph  $H_0 \cup \dots \cup H_{j-1}$  we try to take short hops from  $v = v_0$  to  $v_1$ , from  $v_1$  to  $v_2$ , and so on. A *short* hop is one whose length is at most  $S_\Delta^{j-1}$ . If a short hop exists we call the associated segment *successful*. We will see shortly that  $S_\Delta^{j-1}$  is usually  $\Delta^{j-1} + O(j\Delta^{j-2})$ , which means that  $S_\Delta^{j-1}/\Delta^{j-1}$  tends to 1 as  $\Delta$  (and  $\delta(v, v')$ ) increases. Not all segments will be successful. If we encounter a *failed* segment  $v_\ell \dots v_{\ell+1}$  we require that  $\delta_H(v_\ell, p_j(v_\ell)) = \delta(v_\ell, p_j(v_\ell)) \leq F_\Delta^{j-1}$ , where  $F_\Delta^{j-1}$  is usually  $O(\Delta^{j-1})$ . Suppose at least one segment fails and let  $z$  and  $z'$  be, respectively, the first vertex of the first failed segment and the last vertex of the last failed segment. Let  $s$  and  $s'$  be the number of segments between  $v$  and  $z$ , and between  $v'$  and  $z'$ , respectively. By the definition of failure  $\delta(z, p_j(z))$  and  $\delta(z', p_j(z'))$  are at most  $F_\Delta^{j-1}$  and by the triangle inequality  $\delta(p_j(z), p_j(z')) \leq 2F_\Delta^{j-1} + (\Delta - s - s')\Delta^{j-1}$ . If we could get from  $p_j(z)$  to  $p_j(z')$  by a shortest (or almost shortest) path in  $H_j$  then we would declare the whole path  $v \dots v'$  a success. Whether this is possible depends on whether  $p_j(z')$  lies within  $\mathcal{B}_j(p_j(z))$ . If  $\tau(j) = \mathbf{B}$  we actually require that  $p_j(z')$  be within  $\mathcal{B}_j^{1/2}(p_j(z))$  and if  $\tau(j) = \mathbf{C}$  we would require that  $p_j(z') \in \mathcal{B}_j^{1/3}(p_j(z))$ . In any case, if there is *not* a short path from  $p_j(z)$  to  $p_j(z')$  we have an upper bound on  $\text{rad}_{i+1}(p_j(z)) = O(\delta(p_j(z), p_j(z')))$ , and, therefore, an upper bound on  $\text{rad}_{i+1}(v)$  as well. In this way we can bound  $S_\Delta^j$  and  $F_\Delta^j$  in terms of  $S_\Delta^{j-1}$ ,  $F_\Delta^{j-1}$ , and  $\tau(j)$ . Formally, we define  $S$  and  $F$  with respect to some spanner as follows.

**Definition 1.** Let  $H$  be a spanner defined by some finite string  $\tau \in \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}^*$ . We define  $S_\Delta^j$  and  $F_\Delta^j$  to be minimal such that for any two vertices  $v, v'$  at distance at most  $\Delta^j$ , where  $0 \leq j \leq o$ , at least one of the following inequalities holds:

$$\delta_H(v, v') \leq S_\Delta^j \quad \text{or} \quad \delta_H(v, V_{j+1}) \leq F_\Delta^j$$

Notice that, despite the terminology, a path may both succeed and fail. Any two vertices at distance at most  $\Delta^o$  must be connected in  $H$  by a path of length at most  $S_\Delta^o$ , that is, every such path must be a success. Such a path cannot fail because  $V_{o+1}$  does not exist, and, therefore  $\delta_H(v, V_{o+1})$  is undefined. This simply reflects the fact that in our connection schemes, all vertices in  $V_o$  are connected by (nearly) shortest paths  $H$ .



**Fig. 3.** The vertices  $v$  and  $v'$  are at distance at most  $\Delta^j$ . The vertices  $z$  and  $z'$  are, respectively, the first on the first failed segment and last on the last failed segment.

Lemma [II](#) shows that  $S$  and  $F$  are bounded by some straightforward recurrences. It only considers spanners that employ scheme **A** at the zeroth level, which is generally the wisest choice.

**Lemma 1.** *Consider a spanner defined by  $\tau = \mathbf{A} \cdot \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}^o$ . Then  $S_\Delta^0 = F_\Delta^0 = 1$  holds for all  $\Delta$  and:*

$$F_\Delta^j \leq \begin{cases} 3F_\Delta^{j-1} + \Delta^j & \text{for } \tau(j) = \mathbf{A} \\ 5F_\Delta^{j-1} + 2\Delta^j & \text{for } \tau(j) = \mathbf{B} \\ 7F_\Delta^{j-1} + 3\Delta^j & \text{for } \tau(j) = \mathbf{C} \end{cases}$$

$$S_\Delta^j \leq \max \text{ of } \Delta S_\Delta^{j-1} \text{ and}$$

$$\begin{cases} (\Delta - 1)S_\Delta^{j-1} + 4F_\Delta^{j-1} + \Delta^{j-1} & \text{for } \tau(j) \in \{\mathbf{A}, \mathbf{C}\} \\ (\Delta - 1)S_\Delta^{j-1} + 4F_\Delta^{j-1} + \Delta^{j-1} + 2 & \text{for } \tau(j) = \mathbf{B} \end{cases}$$

*Proof.* For the base case, consider any adjacent  $v, v'$  in  $G$ . If the edge  $(v, v')$  is in  $H_0$  (returned by **A** at level 0) then  $\delta(v, v') = 1 = S_\Delta^0$ . If not then, by the definition of **A**,  $v' \notin \mathcal{B}_0(v)$  and  $\delta_{H_0}(v, p_1(v)) = 1 = F_\Delta^0$ . Let  $v, v', z, z', s$ , and  $s'$  be as in the above discussion; see Figure [3](#). If the spanner does not contain a short path from  $p_j(z)$  to  $p_j(z')$  (failure) then we can conclude that  $p_j(z') \notin \mathcal{B}_j(p_j(z))$  if  $\tau(j) = \mathbf{A}$ , that  $p_j(z') \notin \mathcal{B}_j^{1/2}(p_j(z))$  if  $\tau(j) = \mathbf{B}$ , and that  $p_j(z') \notin \mathcal{B}_j^{1/3}(p_j(z))$  if  $\tau(j) = \mathbf{C}$ . It follows that:

$$\delta(p_j(z), p_{j+1}(p_j(z))) \leq \begin{cases} 2F_\Delta^{j-1} + (\Delta - s - s')\Delta^{j-1} & \text{if } \tau(j) = \mathbf{A} \\ 2(2F_\Delta^{j-1} + (\Delta - s - s')\Delta^{j-1}) & \text{if } \tau(j) = \mathbf{B} \\ 3(2F_\Delta^{j-1} + (\Delta - s - s')\Delta^{j-1}) & \text{if } \tau(j) = \mathbf{C} \end{cases}$$

The distance from  $v$  to  $V_{j+1}$  is at most  $\delta(v, p_{j+1}(p_j(z)))$ , which we bound as:

$$\begin{aligned}
 \delta(v, p_{j+1}(v)) &\leq \delta(v, z) + \delta(z, p_j(z)) + \delta(p_j(z), p_{j+1}(p_j(z))) \\
 &\leq s\Delta^{j-1} + F_{\Delta}^{j-1} + t(2F_{\Delta}^{j-1} + (\Delta - s - s')\Delta^{j-1}) \\
 &\quad \{t = 1, 2, 3 \text{ depending on } \tau(j)\} \\
 &\leq (s + t(\Delta - s - s'))\Delta^{j-1} + (2t + 1)F_{\Delta}^{j-1} \\
 &\leq (2t + 1)F_{\Delta}^{j-1} + t\Delta^j \quad \{\text{worst case is } s = s' = 0\}
 \end{aligned}$$

We obtain the claimed bounds on  $F_{\Delta}^j$  by setting  $t = 1, 2$ , and  $3$  when  $\tau(j)$  is, respectively, **A**, **B**, and **C**. This covers the case when the path  $v \dots v'$  is a failure. One way for it to be a success is if each of the  $\Delta$  segments is a success, that is, if  $z$  and  $z'$  do not exist. In general there will be some failed segments and we can only declare the path successful if there is a *short* path from  $p_j(z)$  to  $p_j(z')$ . We demand a shortest path if  $\tau(j) \in \{\mathbf{A}, \mathbf{C}\}$  and tolerate an additive error of 2 if  $\tau(j) = \mathbf{B}$ . We can now bound  $S_{\Delta}^j$  as follows:

$$\begin{aligned}
 \delta_H(v, v') &\leq \max\{\Delta S_{\Delta}^{j-1}, \delta_H(v, z) + \delta_H(z, p_j(z)) + \delta_H(p_j(z), p_j(z')) \\
 &\quad + \delta_H(p_j(z'), z') + \delta_H(z', v')\} \\
 &\leq \max\{\Delta S_{\Delta}^{j-1}, (s + s')S_{\Delta}^{j-1} + 4F_{\Delta}^{j-1} + (\Delta - s - s')\Delta^{j-1} [+2]\} \\
 &\leq \max\{\Delta S_{\Delta}^{j-1}, (\Delta - 1)S_{\Delta}^{j-1} + 4F_{\Delta}^{j-1} + \Delta^{j-1} [+2]\}
 \end{aligned}$$

where the “[+2]” is only present if  $\tau(j) = \mathbf{B}$ .

Lemma 2 solves these recurrences for spanners that use schemes **A**, **B**, & **C**.

**Lemma 2. (ABC Spanners)** *Consider any spanner  $H$  defined by  $\tau \in \mathbf{A} \cdot \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}^o$ . If  $\Delta \geq 8$  and  $c = 3\Delta/(\Delta - 7)$  then:*

$$F_{\Delta}^j \leq c\Delta^j \quad S_{\Delta}^j \leq \begin{cases} \Delta^j + 4cj\Delta^{j-1} & \text{for } j \leq \Delta \\ (4c + 1)\Delta^j & \text{for } j \geq \Delta \end{cases}$$

Furthermore,  $F_{\Delta}^o = 0$ , that is, if  $\delta(u, v) \leq \Delta^o$  then  $\delta_H(u, v) \leq S_{\Delta}^o$ .

*Proof.* Taking the worst cases from Lemma 1 we have  $F_{\Delta}^j \leq 7F_{\Delta}^{j-1} + 3\Delta^j$  and  $S_{\Delta}^j \leq \max\{\Delta S_{\Delta}^{j-1}, (\Delta - 1)S_{\Delta}^{j-1} + 4F_{\Delta}^{j-1} + \Delta^{j-1} + 2\}$ . One can easily verify by induction that  $F_{\Delta}^j \leq c\Delta^j$ . To bound  $S_{\Delta}^j$  assume inductively that it is at most  $\Delta^j + 4cj\Delta^{j-1} - 1$ , and for  $j \geq \Delta$ , that it is at most  $(4c + 1)\Delta^j - 1$ ; these inequalities clearly hold for  $j = 1$ . First consider the case  $j \leq \Delta$ :

$$\begin{aligned}
 S_{\Delta}^j &\leq \max\{\Delta S_{\Delta}^{j-1}, (\Delta - 1)S_{\Delta}^{j-1} + 4F_{\Delta}^{j-1} + \Delta^{j-1} + 2\} \\
 &\leq \max\{\Delta^j + 4c(j - 1)\Delta^{j-1} - \Delta, \\
 &\quad (\Delta - 1)(\Delta^{j-1} + 4c(j - 1)\Delta^{j-2} - 1) + 4c\Delta^{j-1} + \Delta^{j-1} + 2\} \\
 &\leq \max\{\Delta^j + 4cj\Delta^{j-1} - 1, \\
 &\quad \Delta^j + 4c(j - 1)\Delta^{j-1} + 4c\Delta^{j-1} - (4c(j - 1)\Delta^{j-2} + \Delta + 1)\} \\
 &\leq \Delta^j + 4cj\Delta^{j-1} - 1
 \end{aligned}$$

Notice that for  $j = \Delta$  this bound is precisely  $(4c + 1)\Delta^j - 1$ , which serves as our base case for the bounds on  $S_\Delta^j$  for  $j > \Delta$ :

$$\begin{aligned} S_\Delta^j &\leq \max \{ \Delta S_\Delta^{j-1}, (\Delta - 1)S_\Delta^{j-1} + 4F_\Delta^{j-1} + \Delta^{j-1} + 2 \} \\ &\leq \max \{ (4c + 1)\Delta^j - \Delta, (4c + 1)(\Delta^j - \Delta^{j-1}) + 4c\Delta^{j-1} + \Delta^{j-1} - \Delta + 3 \} \\ &\leq (4c + 1)\Delta^j - 1 \end{aligned}$$

Lemma 2 states that in any spanner generated by some string in  $\mathbf{A} \cdot \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}^o$ , the distortion is given by the function  $f(d) = d + O(od^{1-1/o})$ , provided that  $d$  is at least  $8^o$ . As we will see in Theorem 1,  $o$  can be as large as  $\log_{4/3} \log n$  which means that these spanners have weak guarantees for  $d < 8^{\log_{4/3} \log n} < (\log n)^{7.23}$ . See the full version [18] for spanners that better approximate polylogarithmic distances.

Theorem 1 illustrates some nice sparseness-distortion tradeoffs for spanners composed of schemes  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{C}$ . It only considers those generated by sequences  $\mathbf{ABBC}^{o-2}$ , which turns out to optimize sparseness without significantly affecting the distortion. (In other words,  $\mathbf{ABBBB}$  would be denser than  $\mathbf{ABCC}$  and could only improve lower order terms in the distortion.)

**Theorem 1.** *The spanner generated by  $\mathbf{ABB}$  has  $O(n^{6/5})$  edges and distortion function  $f(d) = d + O(\sqrt{d})$ . The spanner generated by  $\mathbf{ABBC}^{o-2}$  has  $O(on^{1+\nu})$  edges, where  $\nu = (\frac{3}{4})^{o-2} / (7 - 2(\frac{3}{4})^{o-2})$ , and distortion  $d + O(od^{1-1/o} + 8^o)$ .*

*Proof.* Let  $H$  be the spanner defined by  $\mathbf{ABB}$ .  $H$  has on the order of  $n/q_1 + n\sqrt{q_1/q_2} + n\sqrt{nq_2}$  edges, which is  $O(n^{6/5})$  for  $q_1 = n^{-1/5}$  and  $q_2 = n^{-3/5}$ . By Lemma 2, if  $\delta(v, v') \leq \Delta^2$  then  $\delta_H(v, v') \leq S_\Delta^2 = \Delta^2 + O(\Delta)$ . (Recall that such a path cannot fail because every pair of vertices in  $V_2$  is connected by a nearly shortest path.) In the general case let  $H$  be generated by  $\mathbf{AB}^2\mathbf{C}^{o-2}$ , for some  $o \geq 3$ . If  $v$  and  $v'$  are at distance at most  $\Delta^o \geq 8^o$  then by Lemma 2  $\delta(v, v') \leq \min\{\Delta^o + O(o\Delta^{o-1}), O(\Delta^o)\}$ . Thus, for any distance  $d$  (possibly less than  $8^o$ ) the distortion is  $f(d) = d + O(od^{1-1/o} + 8^o)$ . We now choose the sampling probabilities so as to optimize the size of  $H$ . They will be selected so that each of the levels zero through  $o$  contributes about the same number of edges, say  $n^{1+\nu}$ . Since the first three levels contribute  $n/q_1 + n\sqrt{q_1/q_2} + n\sqrt{q_2/q_3}$  edges (scheme  $\mathbf{A}$  at level 0,  $\mathbf{B}$  at 1 and 2), it follows that  $q_1 = n^{-\nu}$ ,  $q_2 = n^{-3\nu}$ , and  $q_3 = n^{-5\nu}$ . Starting from the other end, level  $o$  (scheme  $\mathbf{C}$ ) contributes  $n + n^{2.5}q_o^2$  implying  $q_o = n^{-3/4+\nu/2}$ . For  $3 \leq j < o$ , level  $j$  contributes on the order of  $n + nq_j^2/q_{j+1}^{3/2}$  edges, implying  $q_j = q_{j+1}^{3/4}n^{\nu/2}$ . Assuming inductively that  $q_{j+1} = n^{-(\frac{3}{4})^{o-j+\nu}(2-\frac{3}{2}(\frac{3}{4})^{o-(j+1)})}$  (which holds for the base case  $j + 1 = o$ ), we have, for  $3 \leq j < o$ :

$$q_j = q_{j+1}^{3/4}n^{\nu/2} = n^{-(\frac{3}{4})^{o-j+1} + \frac{3}{4}\nu(2-\frac{3}{2}(\frac{3}{4})^{o-(j+1)}) + \nu/2} = n^{-(\frac{3}{4})^{o-(j-1)} + \nu(2-\frac{3}{2}(\frac{3}{4})^{o-j})}$$

The only sampling probability under two constraints is  $q_3$ , which means that  $\nu$  should be selected to satisfy  $n^{-5\nu} = n^{-(\frac{3}{4})^{o-2} + \nu(2-\frac{3}{2}(\frac{3}{4})^{o-3})}$ . This equality holds for  $\nu = (\frac{3}{4})^{o-2} / (7 - \frac{3}{2}(\frac{3}{4})^{o-3})$ . The size of  $H$  is, therefore, on the order of  $on^{1+\nu}$ .

Let us briefly compare the size bounds obtained above to the spanners of Thorup and Zwick [22]. For distortions  $d + O(\sqrt{d})$ ,  $d + O(d^{2/3})$ , and  $d + O(d^{3/4})$  the spanners of Theorem 1 have sizes on the order of  $n^{6/5}$ ,  $n^{25/22}$ , and  $n^{103/94}$  in contrast to  $n^{4/3}$ ,  $n^{5/4}$ , and  $n^{6/5}$  obtained in [22]. The separation in density becomes sharper as  $o$  increases. For  $o = \log_{4/3} \log n$ , the size and distortion of our spanners is  $O(n \log \log n)$  and  $d + O(od^{1-1/o} + o^o)$ , in contrast to [22], where the size and distortion are  $O(on^{1+1/o})$  and  $d + O(d^{1-1/o} + 2^o)$ . In this case Theorem 1 gives a doubly exponential improvement in density. In some ways Theorem 1 is our strongest result. However, when the order  $o$  is large (close to  $\log_{4/3} \log n$ ) and the distance being approximated very short, the spanners of Theorem 1 cannot guarantee good distortion. See the full version [18] for constructions that address these shortcomings.

### 4 The Connection Schemes

We only analyze schemes **A** and **C**. See [18] for a description of **B**, **D**, and **x**.

*Connection scheme A.* The subgraph returned by **A** at level  $i$  is, by definition,  $\bigcup_{v \in V_i, u \in \bar{\mathcal{B}}_i(v)} P(v, u)$ , that is, a breadth first search tree from every  $v \in V_i$  containing  $p_{i+1}(v)$  and all vertices  $u$  closer to  $v$  than  $p_{i+1}(v)$ . The expected size of this subgraph is at most  $\sum_{v \in V} \Pr[v \in V_i] \cdot \mathbb{E}[|\bar{\mathcal{B}}_i(v)| - 1] \leq nq_i/q_{i+1}$ .

*Connection scheme C.* To analyze scheme **C** we appeal to a lemma of Coppersmith and Elkin [8]. Let  $\mathcal{Q}$  be a set of shortest paths. We say that  $v$  is a *branching point* for two paths  $P, P' \in \mathcal{Q}$  if  $P$  and  $P'$  intersect and  $v$  is an endpoint on the path  $P \cap P'$ . Notice that if  $P$  and  $P'$  have just one vertex in common it would be the unique endpoint on the edgeless path  $P \cap P'$ . Let  $\text{br}(v)$  be the number of pairs  $P, P' \in \mathcal{Q}$  for which  $v$  is a branching point, and let  $\text{br}(\mathcal{Q}) = \sum_{v \in V} \text{br}(v)$ .

**Theorem 2.** (Coppersmith and Elkin) *Let  $\mathcal{Q}$  be a set of shortest paths and  $G(\mathcal{Q}) = \bigcup_{P \in \mathcal{Q}} P$ . Then  $|G(\mathcal{Q})| \leq n + O(\sqrt{n \text{br}(\mathcal{Q})})$ .*

*Proof.* Let  $\text{deg}(v)$  be the degree of  $v$  in  $G(\mathcal{Q})$ . Notice that  $\text{br}(v) \geq \binom{\lceil \text{deg}(v)/2 \rceil}{2}$ . There must be at least  $\lceil \text{deg}(v)/2 \rceil$  paths in  $\mathcal{Q}$  that intersect  $v$ , no two of which use the same edges incident to  $v$ . Each pair of these paths contributes to  $\text{br}(v)$ . We can calculate  $|G(\mathcal{Q})|$  as:  $\frac{1}{2} \sum_v \text{deg}(v) = n + \sum_{v: \text{deg}(v) \geq 3} O(\sqrt{\text{br}(v)}) = n + O(\sqrt{n \text{br}(\mathcal{Q})})$ . The last equality follows from the concavity of square root.

**Theorem 3.** *Let  $\mathcal{Q} = \{P(v, u) : v \in V_i, u \in \bar{\mathcal{B}}_i^{1/3}(v)\}$ . Then  $\mathbb{E}[|G(\mathcal{Q})|] = n + O(nq_i^2/q_{i+1}^{3/2})$ . If  $\mathcal{Q}' = \{P(v, u) : (v, u) \in V_o \times V_o\}$  then  $\mathbb{E}[|G(\mathcal{Q}')|] = n + O(n^{2.5}q_o^2)$ .*

*Proof.* Let  $v, w, v', w' \in V_i$ , where  $v' \in \bar{\mathcal{B}}_i^{1/3}(v)$  and  $w' \in \bar{\mathcal{B}}_i^{1/3}(w)$ . We first argue that if  $P(v, v')$  and  $P(w, w')$  intersect then  $w, w' \in \mathcal{B}_i(v)$ . For any vertex  $w$ ,  $\text{rad}_{i+1}(w) \leq \delta(w, v) + \text{rad}_{i+1}(v)$ . Thus, If  $w$  lies outside  $\mathcal{B}_i(v)$  then  $\bar{\mathcal{B}}_i^{1/3}$



$(w) \cap \mathcal{B}_i^{1/3}(v)$  must be empty. Let  $v_a$  be the  $a$ th farthest vertex from  $v = v_1$ , breaking ties arbitrarily.

$$\begin{aligned} \mathbb{E}[\text{br}(\mathcal{Q})] &\leq 2 \sum_{v \in V, 1 < a < b < c} \Pr[\{v, v_a, v_b, v_c\} \subseteq V_i \wedge \{v_a, v_b, v_c\} \subseteq \mathcal{B}_i(v)] \\ &\leq \sum_{v \in V, c \geq 4} \Pr[|\mathcal{B}_i(v)| \geq c] \cdot c^2 \cdot q_i^4 \leq \sum_{v, c} (1 - q_{i+1})^c \cdot c^2 \cdot q_i^4 = O(nq_i^4/q_{i+1}^3) \end{aligned}$$

The second line follows since  $v_c \in \mathcal{B}_i(v)$  if and only if  $|\mathcal{B}_i(v)| \geq c$ , and once  $v_c$  and  $v = v_1$  are chosen there are  $\binom{c-2}{2}$  ways to choose  $v_a$  and  $v_b$ . The last line follows since  $(1 - q_{i+1})^c$  is bounded by a constant for  $c < 1/q_{i+1}$  and geometrically decaying thereafter. Thus,  $\mathbb{E}[|G(\mathcal{Q})|] = n + O(\sqrt{n \text{br}(\mathcal{Q})}) = n + O(nq_i^2/q_{i+1}^{3/2})$ . Similarly,  $\text{br}(\mathcal{Q}')$  is sharply concentrated around its mean—at most  $(q_{i+1}n)^4$ —and  $\mathbb{E}[|G(\mathcal{Q}')|] = n + O(\mathbb{E}[\sqrt{n \text{br}(\mathcal{Q}')}] = n + O(n^{2.5}q_i^2)$ .

## 5 Conclusion

In this paper we have shown that nearly all the recent work on additive and low distortion spanners can be seen as merely instantiating a generic modular algorithm. The contribution of this work is not only a simpler way to look at spanners. On purely quantitative terms our constructions provide substantially better distortion than [11,22] at any desired level of sparsity. Our constructions can also produce spanners with a linear number of edges. The last construction to achieve linearity was, quite surprisingly, Althöfer et al.'s [2] simple greedy algorithm.

Although the specific tradeoffs of our spanners could certainly be improved, the framework of this paper seems inherently incapable of generating arbitrarily sparse purely additive spanners. It is unclear whether a fundamentally new technique is required to find additive spanners or whether the path-buying algorithm of Baswana et al. [5] could be generalized for this purpose. In any case, proving or disproving the existence of additive spanners remains the chief open problem in this area.

*Acknowledgment.* Thanks to Michael Elkin, Mikkel Thorup, and Uri Zwick for inspiring much of this work.

## References

1. Aingworth, D., Chekuri, C., Indyk, P., Motwani, R.: Fast estimation of diameter and shortest paths. *SIAM J. Comput* 28(4), 1167–1181 (1999)
2. Althöfer, I., Das, G., Dobkin, D., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. *Discrete and Computational Geometry* 9, 81–100 (1993)
3. Awerbuch, B.: Complexity of network synchronization. *J. ACM* 32, 804–823 (1985)
4. Baswana, S., Kavitha, T.: Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In: *FOCS 2006*, (2006)

5. Baswana, S., Kavitha, T., Mehlhorn, K., Pettie, S.: New constructions of  $(\alpha, \beta)$ -spanners and purely additive spanners. In: SODA 2005, (2005)
6. Bollobás, B., Coppersmith, D., Elkin, M.: Sparse subgraphs that preserve long distances and additive spanners. *SIAM J. Discr. Math.* 9(4), 1029–1055 (2006)
7. Coppersmith, D., Elkin, M.: Sparse source-wise and pair-wise distance preservers. In: SODA 2005 (2005)
8. Coppersmith, D., Elkin, M.: Sparse source-wise and pair-wise preservers. *SIAM J. Discrete Math* (to appear)
9. Cowen, L.J., Wagner, C.G.: Compact roundtrip routing in directed networks. *J. Algor.* 50(1), 79–95 (2004)
10. Dor, D., Halperin, S., Zwick, U.: All-pairs almost shortest paths. *SIAM J. Comput.* 29(5), 1740–1759 (2000)
11. Elkin, M., Peleg, D.:  $(1 + \epsilon, \beta)$ -spanner constructions for general graphs. *SIAM J. Comput.* 33(3), 608–631 (2004)
12. Elkin, M., Zhang, J.: Efficient algorithms for constructing  $(1 + \epsilon, \beta)$ -spanners in the distributed and streaming models. In: PODC 2004 (2004)
13. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.* 69(3), 485–497 (2004)
14. Halperin, S., Zwick, U.: Unpublished result 1996
15. Narasimhan, G., Smid, M.: *Geometric Spanner Networks* (2007)
16. Peleg, D., Schaffer, A.A.: Graph spanners. *J. Graph Theory* 13, 99–116 (1989)
17. Peleg, D., Ullman, J.D.: An optimal synchronizer for the hypercube. *SIAM J. Comput.* 18, 740–747 (1989)
18. Pettie, S.: Low distortion spanners. See, <http://www.eecs.umich.edu/~pettie>
19. Roditty, L., Thorup, M., Zwick, U.: Roundtrip spanners and roundtrip routing in directed graphs. In: SODA 2002 (2002)
20. Spielman, D.A., Teng, S.-H.: Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In: STOC 2004 (2004)
21. Thorup, M., Zwick, U.: Compact routing schemes. In: SPAA 2001 (2001)
22. Thorup, M., Zwick, U.: Spanners and emulators with sublinear distance errors. In: SODA 2006 (2006)
23. Thorup, M., Zwick, U.: Approximate distance oracles. *J.ACM* 52, 1–24 (2005)
24. Wenger, R.: Extremal graphs with no  $C^4$ 's,  $C^6$ 's, or  $C^{10}$ 's. *J. Combin. Theory Ser. B* 52(1), 113–116 (1991)
25. Woodruff, D.: Lower bounds for additive spanners, emulators, and more. In: FOCS 2006 (2006)

# Minimum Weight 2-Edge-Connected Spanning Subgraphs in Planar Graphs

André Berger<sup>1</sup> and Michelangelo Grigni<sup>2</sup>

<sup>1</sup> Department of Mathematics, Technical University Berlin, Axel-Springer-Str. 54A,  
10623 Berlin, Germany

`berger@math.tu-berlin.de`

<sup>2</sup> Department of Mathematics and Computer Science, Emory University,  
Atlanta GA 30322, USA

`mic@mathcs.emory.edu`

**Abstract.** We present a linear time algorithm exactly solving the *2-edge connected spanning subgraph (2-ECSS) problem* in a graph of bounded treewidth. Using this with Klein’s diameter reduction technique [15], we find a linear time PTAS for the problem in unweighted planar graphs, and the first PTAS for the problem in weighted planar graphs.

## 1 Introduction

A graph  $G = (V, E)$  is *2-edge-connected* (2-EC) if  $G - e$  is connected for every edge  $e \in E$ . Given  $G$  and a non-negative edge weights  $w \in \mathbb{R}_+^E$ , the *2-edge-connected spanning subgraph (2-ECSS) problem* is to find a 2-ECSS of  $G$  of minimum weight. We consider PTAS (polytime approximation scheme) results. For our purposes, a PTAS is an algorithm taking an instance  $(G, w)$  and  $\varepsilon > 0$  as inputs, running in polynomial time for each fixed  $\varepsilon$ , and finding a solution with weight at most  $1 + \varepsilon$  times optimal.

Like metric TSP, the 2-ECSS problem is NP-hard, even when restricted to unweighted planar graphs, and MAX-SNP-hard for bounded degree graphs [9]. In particular, 2-ECSS is NP-hard to approximate within  $1573/1572$  on graphs of maximum degree 3 [7]. For unweighted graphs the best known polytime algorithm has approximation ratio  $5/4$  [12], and for weighted graphs the best known ratio is 2 [13, 14].

In Section 2 we present an algorithm solving the 2-ECSS problem exactly in linear time for weighted graphs of bounded treewidth. Graphs of bounded treewidth have particular algorithmic interest, since they adapt very well to dynamic programming techniques. Moreover, graphs of small treewidth are also related to fixed parameter algorithms and the bidimensionality theory [10].

A PTAS is known for 2-ECSS in complete geometric graphs of low dimension [9], and for unweighted planar graphs [8]. The latter PTAS runs in time  $n^{O(1/\varepsilon)}$ . We would prefer an “efficient” PTAS (EPTAS); that is, running in time  $f(\varepsilon) \cdot n^c$ , where  $c$  is independent of  $\varepsilon$ . Recent examples of improving a PTAS to an EPTAS are the Euclidean TSP ( $n^{O(1/\varepsilon)}$  [2] to  $O(\varepsilon^{-O(\varepsilon)} + n \log n)$  [18]) and

the weighted planar graph TSP ( $n^{O(1/\varepsilon^2)}$  [1] to  $2^{O(1/\varepsilon^2)} \cdot n$  [15]). In the same vein, we present a linear time EPTAS for the 2-ECSS problem in unweighted planar graphs in Section 3. For this we use the diameter reduction approach, which Klein [15] applied to the metric TSP in weighted planar graphs.

For the 2-ECSS problem in weighted planar graphs, the best previously known polytime approximation ratio is 2, achieved by the algorithm for general weighted graphs [13]. There is also an approximation scheme running in time  $n^{O(\log n \cdot \log(1/\varepsilon)/\varepsilon)}$  [3], a quasi-polynomial. In Section 4 we present the first PTAS for this problem. Again we use ideas developed for the metric TSP; however, our situation is more complicated because it is insufficient to replace the input with a spanner (a subgraph which approximates the original metric). Instead, at each step of the dynamic program, we must consider using a small number of edges which are inessential to the metric. The bounding of this number (Lemma 10) is a key novelty in our approach.

## 2 An Exact Algorithm for Graphs of Bounded Treewidth

In this section we design an exact algorithm for the 2-ECSS problem in weighted graphs of bounded treewidth. We first review basic notions for treewidth.

**Definition 1.** Let  $G = (V, E)$  be a graph. A tree decomposition of  $G$  is a pair  $\mathcal{TD} = (\{X_i : i \in I\}, T = (I, F))$ , where  $\{X_i : i \in I\}$  is a family of subsets of  $V$  and  $T$  is a tree, satisfying:

- (i)  $\bigcup_{i \in I} X_i = V$ ,
- (ii) for any edge  $uv \in E$  there exists an  $i \in I$  such that  $u, v \in X_i$ , and
- (iii) for any  $v \in V$ ,  $T[\{i \in I : v \in X_i\}]$  is a subtree of  $T$ .

The width of the tree decomposition is  $\max_i |X_i| - 1$ . The treewidth  $tw(G)$  of  $G$  is the minimum width of a tree decompositions of  $G$ .

Finding the treewidth of a graph is NP-hard. However, for a given integer  $k$ , one can find a tree decomposition of a graph  $G$  of width at most  $k$  or decide that  $tw(G) > k$  in time  $O(f(k) \cdot n)$  [5]. In [19] it was shown that  $f(k) = 2^{O(k^3)}$ , which makes the algorithm in [5] impractical for large values of  $k$ . For algorithmic purposes a especially simple tree decomposition is very useful.

**Definition 2.** A tree decomposition  $(\{X_i : i \in I\}, T)$  of a graph  $G = (V, E)$  rooted at some  $r \in I$  is called nice, if the following conditions are satisfied:

1. Every node of the tree  $T$  has at most two children.
2. If a node  $i \in I$  has two children  $j$  and  $k$ , then  $X_i = X_j = X_k$  (in this case  $i$  is called a JOIN NODE).
3. If a node  $i$  has one child  $j$ , then either
  - (a)  $|X_i| = |X_j| + 1$  and  $X_j \subseteq X_i$  (in this case  $i$  is called an INSERT NODE),
  - or
  - (b)  $|X_i| = |X_j| - 1$  and  $X_i \subseteq X_j$  (in this case  $i$  is called a FORGET NODE).

**Lemma 1** ([17, Lemma 13.1.3]). *Given a tree decomposition of a graph  $G = (V, E)$  that has width  $k$  and  $O(n)$  nodes, where  $n = |V|$ , we can find a nice tree decomposition of  $G$  that also has width  $k$  and  $O(n)$  nodes in time  $O(kn)$ .*

From now on we assume  $G = (V, E)$  is a weighted graph with weight vector  $w \in \mathbb{R}_+^E$ , and we let  $n = |V|$ . Furthermore, let  $(\{X_i : i \in I\}, (I, F))$  be a nice tree decomposition of  $G$  of width  $k$  and order  $|I| = O(n)$ , rooted at  $r \in I$ . For any  $i \in I$  we define  $V_i$  to be the set of vertices contained in  $X_i$  and all  $X_j$ , for which  $j$  is a descendant of  $i$  in  $T$ . Moreover, we let  $G_i = G[V_i]$ . Note that  $V_r = V$  and  $G_r = G$ .

We now consider the problem of finding a minimum weight 2-ECSS of  $G$  using this tree decomposition. For each  $i \in I$  we want to solve the following subproblem. Given a set of edges  $S \subseteq E(X_i)$  and a graph  $H = (V_H, E_H)$  with  $V_H \cap V = X_i$ , find a subgraph  $G(i, S, H)$  of  $G_i$  of minimum weight, such that  $E(G(i, S, H)) \cap E(X_i) = \emptyset$  and such that  $G(i, S, H) \cup S \cup H$  is 2-edge-connected. We let  $w(i, S, H)$  denote the weight of the graph  $G(i, S, H)$ .

In particular, the graph  $G(r, S^*, H_\emptyset) \cup S^*$ , where  $S^*$  minimizes  $w(r, S, H_\emptyset) + w(S^*)$  among all choices of  $S \subseteq E(X_r)$  ( $H_\emptyset = (X_r, \emptyset)$ ), is a minimum weight 2-ECSS of  $G$ . To determine  $w(r, S, H_\emptyset)$  for  $S \subseteq E(X_r)$  we will work up the tree decomposition from the leaves to the root solving the aforementioned subproblems. The graph  $H$  in a particular subproblem describes how the vertices in  $X_i$  may be connected outside the graph  $G_i$ . We therefore refer to  $H$  as an *external type of  $G_i$* . We can think of  $H$  as a subgraph of  $G[(V \setminus V_i) \cup X_i]$ , that is part of a solution to the 2-ECSS problem on  $G$ , complementing the solution on  $G_i$ .

In general, there is an exponential number of subgraphs of  $G[(V \setminus V_i) \cup X_i]$  that can appear as an external type of the graph  $G_i$ . However, as we will show, the number of external types  $H$  of the graph  $G_i$ , which must be considered for a subproblem at the vertex  $i \in I$  of the tree decomposition, can be bounded only in terms of  $|X_i|$ . Given a graph  $H = (V_H, E_H)$ , and a subset  $X \subseteq V_H$ , there is an algorithm [8] returning a simplified graph  $ext(H, X)$ , which has the same 2-edge-connected properties when used as an external type for another graph, which shares only vertices in  $X$  with  $H$ . The detailed properties of  $ext(H, X)$  are given in Lemma [2] and Corollary [1].

**Lemma 2.** *Let  $H = (V_H, E_H)$  be a graph and  $X \subseteq V_H$ . Moreover, let  $H' = ext(H, X)$ . Then, if  $H^*$  is another graph with  $V_{H^*} \cap V_H = X$ , then  $H^* \cup H$  is 2-EC if and only if  $H^* \cup H'$  is 2-EC. Furthermore,  $|V_{H^*}| \leq 3|X|$  and  $ext(H, X)$  can be found in time linear in  $|V_H| + |E_H|$ .*

Lemma [2] implies that  $G(i, S, H) = G(i, S, ext(H, X_i))$ , and that  $ext(H, X_i)$  does not have too many vertices. To make the computation efficient, we need that the number of possible external types that can appear during the algorithm is small. We let  $\mathcal{T}_{ext}(X) = \{ext(H, X) : H = (V_H, E_H), X \subseteq V_H\}$  denote the set of all possible graphs  $ext(H, X)$  for a fixed set of vertices  $X$ . The following corollary gives us the bound on the number of possible external types.

**Corollary 1.** *For any set  $X$  we have  $|\mathcal{T}_{ext}(X)| = 2^{O(|X| \cdot \log |X|)}$ .*

The algorithm for the 2-ECSS problem will compute the graphs  $G(i, S, H)$  and the values  $w(i, S, H)$  for every possible choice of  $i \in I$ ,  $S \subseteq E(X_i)$  and  $H \in \mathcal{T}_{ext}(X_i)$ , working up from the leaves of  $T$  to the root  $r$ . The following four lemmata will help us prove the correctness of our algorithm.

**Lemma 3.** *Let  $i \in I$  be a leaf in  $T$ . Moreover let  $S \subseteq E(X_i)$  and let  $H \in \mathcal{T}_{ext}(X_i)$ . Then  $G(i, S, H) = \emptyset$  and  $w(i, S, H) = 0$ .*

*Proof.* This is true by the definition of  $G(i, S, H)$ .  $\square$

**Lemma 4.** *Let  $i \in I$  be a join node with children  $j_1$  and  $j_2$ ,  $S \subseteq E(X_i)$  and let  $H \in \mathcal{T}_{ext}(X_i)$ . Then*

$$w(i, S, H) = \min_{H_1 \in \mathcal{T}_{ext}(X_{j_1})} \left[ w(j_1, S, H_1) + w(j_2, S, ext(H \cup G(j_1, S, H_1), X_{j_2})) \right].$$

*Proof.* The idea of the proof is as follows. If  $i$  is a join node with children  $j_1$  and  $j_2$ , then the subgraph  $G(i, S, H)$  of  $G_i$  achieving the minimum weight  $w(i, S, H)$  is a disjoint union of two subgraphs  $G_1$  and  $G_2$  of  $G_{j_1}$  and  $G_{j_2}$ , respectively. We show that then  $G_1$  and  $G_2$  must be solutions to one of the previously computed subproblems at the nodes  $j_1$  and  $j_2$ , related by the formula in the lemma.

Let  $H_1 \in \mathcal{T}_{ext}(X_{j_1})$  be the graph that achieves the minimum value  $W$  on the right hand side of the equality. Furthermore, let  $G_1 = G(j_1, S, H_1)$  and  $G_2 = G(j_2, S, ext(H \cup G_1, X_{j_2}))$ . Now  $G_2 \cup S \cup (H \cup G_1)$  is 2-EC and thus  $G_1 \cup G_2$  is a feasible solution to the subproblem  $(i, S, H)$ . Note that  $G_1$  and  $G_2$  are edge-disjoint, since  $X_{j_1} = X_{j_2}$  and  $G_1$  and  $G_2$  both do not contain any edges between vertices in  $X_{j_1}$ . Hence  $w(i, S, H) \leq w(G_1 \cup G_2) = w(G_1) + w(G_2) = W$ .

To prove the reverse inequality, let  $H_{opt}$  be a subgraph of  $G_i$  with weight  $w(H_{opt}) = w(i, S, H)$  such that  $H_{opt} \cup S \cup H$  is 2-EC. Let  $G_1 = H_{opt}[V_{j_1} \setminus X_{j_1}]$  and  $G_2 = H_{opt}[V_{j_2} \setminus X_{j_2}]$ . Also let  $H_1 = ext(H \cup G_2, X_{j_1})$ . Since  $H_{opt} = G_1 \cup G_2$ , we have that  $G_1 \cup S \cup H_1$  is 2-EC and thus  $w(j_1, S, H_1) \leq w(G_1)$ . Next, we let  $G'_1 = G(j_1, S, H_1)$ , i.e.  $G'_1 \cup S \cup (H \cup G_2)$  is 2-EC. Hence  $G_2$  is a feasible solution to the subproblem  $(j_2, S, ext(H \cup G'_1, X_{j_2}))$  and therefore  $w(j_2, S, ext(H \cup G'_1, X_{j_2})) \leq w(G_2)$ . We finally get  $W \leq w(j_1, S, H_1) + w(j_2, S, ext(H \cup G'_1, X_{j_2})) \leq w(G_1) + w(G_2) = w(H_{opt}) = w(i, S, H)$ .  $\square$

The proofs of the following two lemmas are similar to that of Lemma 4 and we omit them in this abstract.

**Lemma 5.** *Let  $i \in I$  be a insert node with child  $j$ ,  $v \in V \setminus X_j$  such that  $X_i = X_j \cup \{v\}$ ,  $S \subseteq E[X_i]$  and let  $H \in \mathcal{T}_{ext}(X_i)$ . Then*

$$w(i, S, H) = w(j, S \cap E(X_j), ext(H \cup (S \cap E(v, X_j)), X_j)).$$

**Lemma 6.** *Let  $i \in I$  be a forget node with child  $j$ ,  $v \in V$  such that  $X_i = X_j \setminus \{v\}$ ,  $S \subseteq E(X_i)$  and let  $H \in \mathcal{T}_{ext}(X_i)$ . Then*

$$w(i, S, H) = \min_{S_1 \subseteq E(v, X_j)} \left[ w(j, S \cup S_1, H_1 = (V_H \cup \{v\}, E_H)) + w(S_1) \right].$$

Using Lemmas 3.4, 3.6, the values  $w(i, S, H)$  as well as the graphs  $G(i, S, H)$  can be computed by using the previously computed values at the children of  $i$  in the tree decomposition.

We now analyze the running time of the algorithm. The tree decomposition has  $O(n)$  vertices and for each  $i \in I$  we have  $2^{k^2} k^{O(k)}$  subproblems, since there are at most  $k^2$  edges in  $E(X_i)$  and at most  $k^{O(k)}$  external types for  $X_i$  (Lemma 3.1). The most time consuming type of subproblem to solve is the one described in Lemma 4.1, which takes  $2^{O(k \log k)}$  time. Thus the total running time of the algorithm is  $O(2^{O(k^2)} \cdot n)$ .

**Theorem 1.** *The 2-ECSS problem in a weighted graph with  $n$  vertices and a tree decomposition of width  $k$  can be solved exactly in  $O(2^{O(k^2)} \cdot n)$  time.*

The constant in the running time of the algorithm can be improved using the following observation. Since  $G(i, S, H)$  is a graph with no edges in  $E(X_i)$ , we have that  $G(i, S, H) = G(i, S', H)$  for all  $S'$  that we can obtain from  $S$  by removing chords. The detailed arguments for this claim are identical to those in the proof of Lemma 2.1. Therefore, during the whole algorithm, we only have to consider edge sets  $S \subseteq E(X_i)$  for a given subproblem  $(i, S, H)$ , which have no chords. We state the following lemma without proof.

**Lemma 7.** *Let  $G = (V, E)$  be a graph with no chords. Then  $|E| \leq 2|V| - 2$ .*

For a given  $i \in I$ , therefore, we need to consider at most  $(k^2)^k = 2^{O(k \log k)}$  different subsets  $S \subseteq E(X_i)$  and we obtain the following improved running time.

**Theorem 2.** *Let a 2-EC graph  $G = (V, E)$  be given with a tree decomposition of width  $k$ , and let  $w \in \mathbb{R}_+^E$  be a non-negative weight vector. Then, in time  $O(2^{O(k \log k)} \cdot n)$ , one can find a 2-ECSS of  $G$  of minimum weight.*

### 3 A Linear-Time PTAS for Unweighted Planar Graphs

We will now apply the methods from the previous section to obtain a linear time PTAS for the 2-ECSS problem in unweighted planar graphs. (More generally, the approach here would also work in weighted planar graphs whenever  $w(G) = O(\text{OPT})$ .)

As in [8], we apply cycle contractions. Suppose  $G = (V, E)$  is a graph,  $C$  is a simple cycle in  $G$ , and  $G/C$  is the result of contracting  $C$  to a vertex (we remove any self-loops, but leave parallel edges). First observe that  $G$  is 2-EC iff  $G/C$  is 2-EC. If  $H'$  is a 2-ECSS in  $G$ , then its image  $(H' \cup C)/C$  is a 2-ECSS in  $G/C$ . Conversely, if  $H$  is a 2-ECSS in  $G/C$ , we can lift it back to a 2-ECSS  $H' = H \cup C$  in  $G$  (so  $H'/C = H$ ). Thus if we can find a near-optimal 2-ECSS  $H$  in  $G/C$ , then we also have a near-optimal 2-ECSS  $H'$  in  $G$ , with at most  $w(C)$  more additive error. Furthermore, if the vertex  $v$  representing  $C$  in  $G/C$  is a cut vertex, then the 2-ECSS problem in  $G/C$  decomposes into independent subproblems, one per block around  $v$ .

The strategy to design the PTAS will be as follows. Given an unweighted planar 2-EC graph  $G$ , let  $\text{OPT}$  denote the size of a minimum 2-ECSS of  $G$ . We will first use methods from [4] and [15] to decompose the graph into parts of small treewidth by contracting a set of low weight cycles and committing only a small error to the solution. The 2-ECSS problems on the parts of small treewidth will be solved exactly using the algorithm from Section 2. So, let  $G = (V, E)$  be an embedded planar 2-EC graph. Our algorithm will make use of the following lemma, which can be derived from methods in [15] and [4].

**Lemma 8.** *Let  $G = (V, E)$  be an embedded planar graph,  $w \in \mathbb{R}_+^E$ , and  $k$  a positive integer. Then one can find a set of simple, edge-disjoint cycles  $\mathcal{C}$ , such that  $w(E(\mathcal{C})) \leq w(G)/k$  and such that the 2-ECSS problem on  $G/\mathcal{C}$  can be decomposed into independent 2-ECSS problems on  $k$ -outerplanar graphs.*

Algorithm 1 gives a general overview of the PTAS for the unweighted planar graph 2-ECSS problem. The subproblems considered in step 2 are 2-EC planar graphs of outerplanarity  $k$ . Let  $H$  be such a graph. Since  $H$  is  $k$ -outerplanar, it can be triangulated in such a way that the outerplanarity of  $H^\Delta$  is still  $k$  and such that every vertex of  $H^\Delta$  has distance at most  $k$  to some vertex on the infinite face. Let  $H^\Delta$  be the triangulated graph. By choosing an arbitrary  $v_0$  on the infinite face and doing a breadth-first search on  $H^\Delta$ , we can obtain a spanning tree  $T$  of  $H^\Delta$  of radius at most  $k$ . This spanning tree will help us define a tree decomposition of  $H^\Delta$ . We let  $I$  be the set of triangles in  $H^\Delta$  and two triangles  $i, j \in I$  form an edge  $ij \in F$  if and only if they share an edge in  $E(H^\Delta) \setminus E(T)$ . Moreover, for a triangle  $i = \{u, v, w\}$ , we let  $X_i$  be the set of all vertices on the three paths from  $u, v$ , and  $w$ , respectively, to the root  $v_0$  of  $T$ . It is shown in [11], that  $\mathcal{TD} = (\{X_i : i \in I\}, (I, F))$  is indeed a tree decomposition of  $H^\Delta$  of width at most  $3k$ , and that it can be found time  $O(kn)$ . Since  $H$  is a subgraph of  $H^\Delta$ ,  $\mathcal{TD}$  is also a tree decomposition of  $H$ . We can now apply the 2-ECSS algorithm for graphs of bounded treewidth to  $H$  and  $\mathcal{TD}$  and we obtain the following theorem.

**Theorem 3.** *Given an unweighted 2-EC planar graph  $G = (V, E)$  and  $\varepsilon > 0$ , Algorithm 1 finds an  $(1 + \varepsilon)$ -approximate 2-ECSS of  $G$  in time  $2^{O(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})} \cdot n$ , where  $n = |V|$ .*

*Proof.* We first prove the approximation guarantee of the algorithm. In any graph a 2-ECSS must use at least  $n$  edges, and therefore  $\text{OPT} \geq n$ . Moreover, since  $G$  is planar,  $|E| \leq 3n - 6$ . The only error committed by the algorithm is adding the edges in  $E(\mathcal{C})$ , for which by Lemma 8 we have  $|E(\mathcal{C})| \leq |E|/k \leq (3n - 6)/\lceil \frac{3}{\varepsilon} \rceil \leq \varepsilon \cdot n \leq \varepsilon \cdot \text{OPT}$ . The running time is dominated by the application of the algorithm from Section 2, and is at most  $2^{O(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})} \cdot n$  by Theorem 2.  $\square$

Moreover, for planar graphs we can derive a better bound on the number of external types that can occur during the dynamic program (cf. [8]). We conclude this section with the following theorem.

**Theorem 4.** *Given an unweighted 2-EC planar graph  $G = (V, E)$  and  $\varepsilon > 0$ , one can find a  $(1 + \varepsilon)$ -approximate 2-ECSS of  $G$  in time  $2^{O(1/\varepsilon)} \cdot n$ , where  $n = |V|$ .*



**Input:** An embedded planar 2-EC graph  $G = (V, E)$ ,  $\varepsilon > 0$ .

**Output:** A  $(1 + \varepsilon)$ -approximate 2-ECSS of  $G$ .

- 1: Apply Lemma 8 to  $G$  with  $k = \lceil 3/\varepsilon \rceil$  and obtain a set of cycles  $\mathcal{C}$ .
- 2: **for all** 2-EC components  $H$  of  $G/\mathcal{C}$  **do**
- 3:     Find a tree decomposition  $\mathcal{TD}$  of  $H$  of width at most  $3k$  as described above.
- 4:     Transform  $\mathcal{TD}$  to a nice tree decomposition using Lemma 11
- 5:     Apply the algorithm from Section 2 to  $H$  and  $\mathcal{TD}$  and obtain  $S_H^*$ .
- 6: **end for**
- 7: **return** the solutions  $S_H^*$  and  $E(\mathcal{C})$ .

**Algorithm 1.** A PTAS for the 2-ECSS problem on unweighted planar graphs

## 4 A PTAS for Weighted Planar Graphs

In this section we will sketch the PTAS for the 2-ECSS problem in weighted planar graphs. The main idea of the algorithm is similar to Algorithm 1, the PTAS for unweighted planar graphs. If Algorithm 1 is applied to a weighted graph  $G$  with weights  $w \in \mathbb{R}_+^E$ , then it will still run in linear time, however the approximation ratio will be  $1 + O(\varepsilon \cdot w(G)/OPT)$ , which may be unbounded for general weights.

To avoid this problem we first construct a spanning subgraph of the input graph whose weight is at most constant times the optimal solution  $OPT$ . Similarly to the algorithm for unweighted planar graphs, we would like to decompose the spanner according to Lemma 8 into independent 2-ECSS problems of bounded outerplanarity, and solve these using a dynamic program similar to the algorithm from Section 2. However, we have to consider that certain edges that were deleted during the construction of the spanner have to appear in any  $(1 + \varepsilon)$ -approximate solution.

In the following we will concentrate on the construction of the desired spanning subgraph. For a weighted graph  $G = (V, E)$ , a weight vector  $w \in \mathbb{R}_+^E$  and some  $s \geq 1$ , an  $s$ -spanner of  $G$  is a spanning subgraph  $H$  of  $G$ , such that for any two vertices  $u, v \in V$  we have  $\text{dist}_H \leq s \cdot \text{dist}_G(u, v)$ , i.e. in  $H$  the distances between any two vertices are approximately the same as the respective distance in  $G$ .

Of course  $G$  itself is an  $s$ -spanner of  $G$  for any  $s \geq 1$ , but in many applications one seeks to find spanners with particular properties, such as few edges or small weight, so that instead of solving the given problem on  $G$  one can instead solve the problem on a much smaller graph. One example for this is the metric-TSP PTAS for weighted planar graphs, which starts with replacing the input graph by an  $s$ -spanner while effectively bounding  $w(G)/OPT$  for the remainder of the algorithm.

Suppose now that  $G$  is a weighted embedded planar graph and  $H$  is a subgraph. A *chord*  $e$  of  $H$  is an edge of  $G$  not in  $H$ . Note that  $H$  and  $e$  inherit embeddings from  $G$ . For each chord  $e$  we define  $w_H(e)$  as the length of the shortest walk connecting the endpoints of  $e$  along the boundary of the face of  $H$  containing  $e$ . The spanner algorithm  $\text{Augment}(G, w, A, S)$  from [3] has as inputs

a weighted planar graph  $G = (V, E)$ , weights  $w \in \mathbb{R}_+^E$ , a subgraph  $A$  of  $G$  and some  $s \geq 1$ . Its properties are as follows:

**Lemma 9** ([3, Theorem 2]). *Let  $H = \text{Augment}(G, w, A, s)$ , where  $G, w, A$  and  $s$  are as above. Then  $H$  is a  $s$ -spanner of  $G$  containing  $A$ . In particular,  $w_H(e) \leq s \cdot w(e)$  for any  $e \in E \setminus E(H)$ . Moreover, if  $A$  is connected, then  $w(H) \leq (1 + 2/(s - 1)) \cdot w(A)$ .*

Given a face  $f$  in  $H$ , the chords of  $f$  are the edges of  $G$  embedded inside  $f$ , according to  $G$ 's embedding. A *face-edge*  $e$  of  $f$  is an abstract edge connecting two vertices of  $f$ ; unlike a chord, a face-edge is not necessarily an edge of  $G$ . (If vertices appear more than once on  $f$ , we must specify which appearances we want as the endpoints of  $e$ .) We say the face edge  $e$  *crosses* a chord  $c$  if  $c$  is a chord of the same face  $f$  and their endpoints are distinct vertex appearances on  $f$ , and they appear in cyclic “*eecc*” order around the boundary of  $f$ . Note that we may embed  $e$  inside  $f$  so  $e$  intersects only the crossed chords. The following lemma shows that the spanner has nice intersection properties with a near-optimal solution to the 2-ECSS problem; it is key in bounding the size of the dynamic program described later.

**Lemma 10.** *Let  $G = (V, E)$  be a planar graph and  $w \in \mathbb{R}_+^E$ . Let  $A$  be a 2-approximate 2-ECSS of  $G$  and  $H = \text{Augment}(G, w, A, \sqrt{2})$ . Furthermore, let  $S^*$  be an optimal 2-ECSS of  $G$  and  $D$  a positive integer.*

*Then there exists a 2-ECSS  $S$  of  $G$  with the following properties:*

1.  $w(S) \leq (1 + \sqrt{2}/D) \cdot w(S^*)$
2. *For any face  $f$  of  $H$ , and any face-edge  $e$  of  $f$ , the number of edges in  $S$  crossing  $e$  is at most  $2D$ .*

*Proof.* Consider a face  $f$  of  $H$  and let  $C$  be the set of chords of  $f$  which are also edges of  $S^*$ . Moreover, let  $E_f$  be the set of edges on the boundary of  $f$ . There are potentially very many edges in  $S^*$  crossing a particular face-edge of  $f$ . We will remove some edges from  $S^*$  and replace them by walks on  $E_f$  as to maintain a 2-edge-connected subgraph  $S$  of  $G$ , while not increasing the weight of  $S^*$  by too much and ensuring the bound on the number of edges in  $S$  crossing any face-edge of  $f$ .

Let  $S_f^*$  be the set of edges in  $S^*$  embedded inside the face  $f$ , i.e. these are edges which are not in  $H$ . The dual edges of all edges in  $S_f^*$  form a tree  $T^*$ , and we will orient each such dual edge in the following way. Any  $e = uv \in S_f^*$  divides the boundary of  $f$ , i.e.  $E_f$ , into two walks  $P_1$  and  $P_2$ , and we will orient the dual edge  $e^*$  towards that part of the boundary of  $f$  which has smaller total weight, and we will call this walk  $P_e$ . Using the definition above this means  $w_H(e) = w(P_e)$ . In particular we know by Lemma 9,  $w(P_e) \leq \sqrt{2} \cdot w(e)$ . Moreover, any vertex of  $T^*$  has indegree at most 1, since if a vertex had indegree at least two, then two disjoint parts of  $E_f$  would both have total weight greater than  $w(E_f)/2$ , which is a contradiction. Therefore there must exist a root vertex  $r$  of  $T^*$ , such that all edges in  $T^*$  are oriented away from  $r$ .

We will now modify  $S^*$  as follows. Let  $T_D^*$  be the set of dual edges at distance  $D$  from  $r$  in  $T^*$ . For any  $e^* \in T_D^*$ , we will delete from  $S^*$  all edges which are dual to the edges descendant to  $e^*$  in  $T^*$ . Moreover, we add  $P_e$  to  $S^*$ . We call the resulting graph  $S$ . Note that  $P_e \cap P_{\bar{e}} = \emptyset$  for any two edges  $e^*, \bar{e}^* \in T_D^*$ . It is clear that  $S$  remains a 2-ECSS of  $G$ . Further, the dual tree of edges in the newly constructed  $S$  which are embedded in  $f$  has now depth at most  $D$ . This shows that any face-edge of  $f$  is now crossed by at most  $2D$  edges in  $S$ .

It remains to show that adding the walks  $P_e$  for all  $e^* \in T_D^*$  increases the weight of  $S^*$  by at most  $\sqrt{2}/D \cdot w(S_f^*)$ . If  $e^* \in T_D^*$  and an edge  $\tilde{e}^*$  is on the path from  $e^*$  to  $r$  in  $T^*$ , we say  $\tilde{e}^* \leq e^*$ . For any fixed  $\tilde{e}^* \in E(T^*)$  of distance at most  $D$  from  $r$  we then have that  $\bigcup_{e^* \in T_D^*: \tilde{e}^* \leq e^*} P_e \subseteq P_{\tilde{e}^*}$ . Thus  $\sum_{e^* \in T_D^*: \tilde{e}^* \leq e^*} w(P_e) \leq w(P_{\tilde{e}^*})$ . In turn, for any fixed  $e^* \in T_D^*$  there are exactly  $D$  edges  $\tilde{e}^*$  with  $\tilde{e}^* \leq e^*$ . Now the total weight by which we increase the weight of the solution  $S^*$  is  $\sum_{e^* \in T_D^*} w(P_e) \leq 1/D \cdot \sum_{e^* \in T_D^*} \sum_{\tilde{e}^* \leq e^*} w(P_e) \leq 1/D \cdot \sum_{\tilde{e}^* \in T^*} \sum_{e^* \in T_D^*: \tilde{e}^* \leq e^*} w(P_e) \leq 1/D \cdot \sum_{\tilde{e}^* \in T^*} w(P_{\tilde{e}^*}) \leq \sqrt{2}/D \cdot w(S_f^*)$ . When doing this procedure for all faces of  $H$ , we obtain the desired 2-ECSS  $S$  of  $G$ . For each face  $f$  of  $H$  we added edges to  $S^*$  of weight at most  $\sqrt{2}/D \cdot w(S_f^*)$ , so the total weight added to  $S^*$  for all faces of  $H$  is at most  $\sqrt{2}/D \cdot w(S^*)$ .  $\square$

Algorithm 2 gives a high-level description of the PTAS for the weighted 2-ECSS problem in planar graphs. Algorithm 3 is a dynamic program based on the ideas from Section 2 and is discussed more thoroughly below. In order for Algorithm 2

**Input:** A planar 2-EC graph  $G = (V, E)$ , edge weights  $w \in \mathbb{R}_+^E$ ,  $\varepsilon > 0$ .  
**Output:** A  $(1 + \varepsilon)$ -approximate 2-ECSS of  $G$ .

- 1: Find a 2-approximate 2-ECSS  $A$  of  $G$  (e.g. using [13]).
- 2:  $H = \text{Augment}(G, \sqrt{2}, A)$
- 3: Apply Lemma 3 to  $H$  with  $k = \lceil \frac{24}{\varepsilon} \rceil$  and obtain a set of cycles  $\mathcal{C}$ .
- 4: **for all** 2-EC components  $F = (V_F, E_F)$  of  $H/\mathcal{C}$  **do**
- 5:     Apply Algorithm 3 to  $F$  with  $D = \lceil \frac{4}{\varepsilon} \rceil$  and obtain  $S_F$ .
- 6: **end for**
- 7: **return** the union of the solutions  $S_F$  and  $E(\mathcal{C})$ .

**Algorithm 2.** A PTAS for the 2-ECSS problem on weighted planar graphs

to be a PTAS for the 2-ECSS problem on weighted planar graphs, we have to make use of Lemma 10, which guarantees that there exists a 2-ECSS of  $G$  of weight at most  $(1 + \sqrt{2}/D) \cdot OPT \leq (1 + \varepsilon/2) \cdot OPT$ , which crosses each face edge of the spanner  $H$  at most  $2D$  times. Algorithm 3 finds a minimum weight 2-ECSS of  $G$  which satisfies this condition for only a certain subset of all face-edges, and therefore it also has weight at most  $(1 + \varepsilon/2) \cdot OPT$ .

**Theorem 5.** *Given a planar 2-EC graph  $G = (V, E)$  and  $\varepsilon > 0$ , Algorithm 2 computes an  $(1 + \varepsilon)$ -approximate 2-ECSS of  $G$  in time  $n^{O(1/\varepsilon^2)}$ , where  $n = |V|$ .*

*Proof.* Let us first prove the approximation guarantee of the algorithm. By Lemma 9 we have that  $w(H) \leq (1 + 1/(\sqrt{2} - 1)) \cdot w(A) \leq 12 \cdot OPT$ , since  $A$  is a 2-approximate 2-ECSS of  $G$ . Applying Lemma 8 in step 3 with  $k = \lceil \frac{24}{\varepsilon} \rceil$  yields a collection of cycles of weight at most  $w(H)/k \leq 12 \cdot OPT / \lceil \frac{24}{\varepsilon} \rceil \leq \frac{\varepsilon}{2} \cdot OPT$ . Therefore, by committing the cycles in  $\mathcal{C}$  to our solution, we add an error of at most  $\frac{\varepsilon}{2} \cdot OPT$  to the solution.

For each 2-EC component  $F$  of  $H/\mathcal{C}$ , we find a  $(1 + \varepsilon/2)$ -approximate 2-ECSS of  $F$  as argued above, and all the 2-ECSS problems on those 2-EC components are independent as described in Section 3. In step 7 we combine the solutions and their total weight is at most  $(1 + \varepsilon/2) \cdot OPT$ . Together with  $\mathcal{C}$  they comprise a 2-ECSS of the original input graph  $G$ , whose total cost is at most  $(1 + \varepsilon) \cdot OPT$ .

The overall running time is dominated by the application of Algorithm 3 in steps 4-6 and is bounded by  $n^{O(kD)} = n^{O(1/\varepsilon^2)}$ .  $\square$

We will now sketch the ideas for Algorithm 3. Let  $F = (V_F, E_F)$  be a 2-EC component of  $H/\mathcal{C}$ . We denote by  $C_F$  the set of edges from the original input graph  $G$  which were deleted during the construction of the spanner  $H$  and are now “missing” from  $F$ . Note that  $F \cup C_F$  is still a planar graph.

We can triangulate each face of  $F$  (as described in Section 3), to obtain a planar graph  $F^\Delta$  and a spanning tree  $T$  of  $F^\Delta$  of radius at most  $k$ . Moreover, let  $E^\Delta = E(F^\Delta) \setminus E_F$ . Some of the edges in  $T$  may be edges in  $E_F$  and some may be edges in  $E^\Delta$ , and we let  $T^\Delta = E(T) \cap E^\Delta$ . We will make use of the same tree decomposition of  $F^\Delta$  as used in the PTAS for unweighted planar graphs. For a vertex  $u \in V_F$  let  $P_u$  denote the path from  $u$  to the root  $v_r$  of  $T$ , and for a triangle  $i = \Delta uvw$ , we denote by  $\mathcal{P}_i = P_u \cup P_v \cup P_w$  the union of the three paths from  $u$ ,  $v$  and  $w$  to  $v_r$ . Remember, we can obtain a tree decomposition  $\mathcal{TD} = (\{X_i : i \in I\}, T_{\mathcal{TD}})$  of  $F^\Delta$  by letting  $I$  be the set of triangles in  $F^\Delta$  and by letting  $X_i = V(\mathcal{P}_i)$  for each  $i \in I$ . Two triangles  $i, j \in I$  are connected by an edge in  $T_{\mathcal{TD}}$  if and only if they share an edge which is not in  $T$ .  $\mathcal{TD}$  is a tree decomposition of  $F^\Delta$  and therefore also of  $F$  of width at most  $3k$ , and every  $i \in I$  has degree at most 3 in  $T_{\mathcal{TD}}$ . We apply Lemma 11 to  $\mathcal{TD}$  and change it to a nice tree decomposition, then each  $X_i$  ( $i \in I$ ) is still the vertex set of the union of three paths  $P_u, P_v$  and  $P_w$ , but now  $u, v$  and  $w$  do not necessarily lie on a triangle of  $F^\Delta$ .

We now define the subproblems that we want to solve in our dynamic program. For an edge  $e \in T^\Delta$ , let  $C_e \subseteq C_F$  be the set of edges in  $C_F$  crossing  $e$ . Let now  $i \in I$  be a node of the tree decomposition. The set of edges in  $C_F$  crossing any edge in  $\mathcal{P}_i$  is denoted by  $C_i = \bigcup_{e \in \mathcal{P}_i} C_e$ . Moreover, we let  $S \subseteq E_F(X_i)$  be a set

of edges of  $F$  with both endpoints in  $X_i$ . Furthermore, let  $C \subseteq C_i$  be a set of crossing edges, such that  $|C \cap C_e| \leq 2D$  for any  $e \in \mathcal{P}_i \cap T^\Delta$ . Finally, for any subproblem we also have to define an external type on the vertices in  $X_i$  and  $V(C)$ , the endpoints of edges in  $C$ . Thus let also a  $\hat{H} \in \mathcal{T}_{ext}(X_i \cup V(C))$  be given.

The subproblem  $(i, S, C, \hat{H})$  is now the following. Find a subgraph  $F^* = F(i, S, C, \hat{H})$  of  $F[V_i] \cup C_F$  of minimum weight  $w(i, S, C, \hat{H})$ , such that  $F^* \cup H$  is

2-edge-connected,  $F^* \cap E_F(X_i) = S$  and such that  $F^* \cap C_i = C$ . If we let  $(S^*, C^*) = \operatorname{argmin}\{w(r, S, C, \hat{H}_\theta) : S \subseteq E_F(X_r), C \subseteq C_i : |C \cap C_e| \leq 2D \text{ for all } e \in \mathcal{P}_r \cap T^\Delta\}$ , then  $F(r, S^*, C^*, \hat{H}_\theta)$  is a minimum weight 2-ECSS of  $F \cup C_F$  which crosses each edge in  $\mathcal{P}_r \cap T^\Delta$  at most  $2D$  times. Therefore the weight of that solution, i.e.  $w(r, S^*, C^*, \hat{H}_\theta)$ , is at most the weight of a minimum weight 2-ECSS of  $F \cap C_F$  which crosses all edges in  $T^\Delta$  at most  $2D$  times. This is the property we needed in the proof of Theorem 5.

Computing the values  $w(i, S, C, \hat{H})$  and the graphs  $F(i, S, C, \hat{H})$  can be done using Lemmas similar to Lemmas 346. The running time is  $n^{O(kD)}$ ; we omit its analysis, except to remark it is dominated by considering all subsets  $C$  of  $O(kD)$  crossing edges.

**Input:** A planar graph  $F$ , a set of edges  $C_F$  and  $D$  (as described above).

**Output:** A 2-ECSS of  $F \cup C_F$ .

- 1: Triangulate  $F$  and obtain  $F^\Delta$  of radius  $\leq k$ .
- 2: Find a spanning tree  $T$  of  $F^\Delta$  of radius  $\leq k$ .
- 3: Find a tree decomposition  $\mathcal{TD}$  of  $F^\Delta$  of width at most  $3k$  using  $T$ .
- 4: Refine  $\mathcal{TD}$  and obtain  $\mathcal{TD}$ , a nice tree decomposition with root  $r$ .
- 5: Solve the dynamic program on  $\mathcal{TD}$  as described above.
- 6:  $(S^*, C^*) = \operatorname{argmin}\{w(r, S, C, \hat{H}_\theta) : S \subseteq E_F(X_r), C \subseteq C_i : |C \cap C_e| \leq 2D \text{ for all } e \in \mathcal{P}_r \cap T^\Delta\}$ .
- 7: **return**  $F(r, S^*, C^*, \hat{H}_\theta)$

**Algorithm 3.** Generalized 2-ECSS for planar graphs

## 5 Future Work

The PTAS results of this paper are likely to carry over to the 2-VCSS problem (minimum weight 2-vertex-connected spanning subgraph), as happened before [8]. A question motivated by the TSP is whether we can find a PTAS for the ‘‘Steiner’’ version of the 2-ECSS problem. That is, along with the graph  $G$ , we are given a subset  $R$  of ‘‘required’’ vertices; we must find a 2-EC subgraph spanning at least the vertices in  $R$ . For the subset version of planar metric TSP, Klein’s subset spanner construction [16] yields an EPTAS. A similar subset spanner construction yields another EPTAS for the Steiner tree problem in planar graphs [6]. We can apply Klein’s subset spanner construction to generalize our Lemma 9; however, the generalization of Lemma 10 remains a roadblock.

## References

1. Arora, S., Grigni, M., Karger, D., Klein, P., Woloszyn, A.: A polynomial-time approximation scheme for weighted planar graph TSP. In: Proceedings of the Ninth Annual ACM-SIAM SODA, pp. 33–41. ACM Press, New York (1998)
2. Arora, S.: Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In: Proceedings of the Thirtyseventh Annual FOCS, pp. 2–11. IEEE Computer Society Press, Los Alamitos (1996)

3. Berger, A., Czumaj, A., Grigni, M., Zhao, H.: Approximation Schemes for Minimum 2-Connected Spanning Subgraphs in Weighted Planar Graphs. In: Brodal, G.S., Leonardi, S. (eds.) *ESA 2005*. LNCS, vol. 3669, pp. 472–483. Springer, Heidelberg (2005)
4. Berger, A., Grigni, M., Zhao, H.: A well-connected separator for planar graphs. Technical Report TR-2004-026-A, Emory University (2004)
5. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing* 25, 1305–1317 (1996)
6. Borradaile, G., Kenyon-Mathieu, C., Klein, P.: A Polynomial-Time Approximation Scheme for Steiner Tree in Planar Graphs. In: *Proceedings of the Eighteenth Annual ACM-SIAM SODA*, ACM Press, New York (2007)
7. Csaba, B., Karpinski, M., Krysta, P.: Approximability of dense and sparse instances of minimum 2-connectivity, TSP and path problems. In: *Proceedings of the 13th Annual ACM-SIAM Symposium On Discrete Mathematics (SODA)*, pp. 74–83. ACM Press, New York (2002)
8. Czumaj, A., Grigni, M., Sissokho, P., Zhao, H.: Approximation schemes for minimum 2-edge-connected and biconnected subgraphs in planar graphs. In: *Proceedings of the Fifteenth Annual ACM-SIAM SODA*, pp. 489–498. ACM Press, New York (2004)
9. Czumaj, A., Lingas, A.: On approximability of the minimumcost  $k$ -connected spanning subgraph problem. In: *Proceedings of the Tenth Annual ACM-SIAM SODA*, pp. 281–290. ACM Press, New York (1999)
10. Demaine, E.D., Hajiaghayi, M.: Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth. In: Pach, J. (ed.) *GD 2004*. LNCS, vol. 3383, pp. 517–533. Springer, Heidelberg (2005)
11. Eppstein, D.: Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications* 3(3), 1–27 (1999)
12. Jothi, R., Raghavachari, B., Varadarajan, S.: A  $5/4$ -approximation algorithm for minimum 2-edge-connectivity. In: *Proceedings of the Fourteenth Annual ACM-SIAM SODA*, pp. 725–734. ACM Press, New York (2003)
13. Khuller, S.: Approximation algorithms for finding highly connected subgraphs. In: Dorit, S. (ed.) *Approximation Algorithms for NP-hard Problems*, PWS Publishing, Company (1996)
14. Khuller, S., Vishkin, U.: Biconnectivity approximations and graph carvings. *Journal of the ACM* 41(2), 214–235 (1994)
15. Klein, P.N.: A linear-time approximation scheme for planar weighted TSP. In: *Proceedings of the 46th Annual IEEE FOCS*, pp. 647–657. IEEE Computer Society Press, Los Alamitos (2005)
16. Klein, P.N.: A subset spanner for planar graphs, with application to subset TSP. In: *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*, pp. 749–756. ACM Press, New York (2006)
17. Kloks, T. (ed.): *Treewidth*. LNCS, vol. 842. Springer, Berlin (1994)
18. Rao, S.B., Smith, W.D.: Approximating geometrical graphs via “spanners” and “banyans”. In: *Proceedings of the Thirtieth annual ACM Symposium on Theory of Computing (STOC 1998)*, pp. 540–550. ACM Press, New York (1998)
19. Röhrig, H.: Tree decomposition: A feasibility study. Master’s thesis, Max-Planck-Institut für Informatik in Saarbrücken (1998)

# Labeling Schemes for Vertex Connectivity

Amos Korman\*

Information Systems Group, Faculty of IE&M, The Technion, Haifa, 32000 Israel  
pandit@tx.technion.ac.il

**Abstract.** This paper studies labeling schemes for the vertex connectivity function on general graphs. We consider the problem of labeling the nodes of any  $n$ -node graph in such a way that given the labels of two nodes  $u$  and  $v$ , one can decide whether  $u$  and  $v$  are  $k$ -vertex connected in  $G$ , i.e., whether there exist  $k$  vertex disjoint paths connecting  $u$  and  $v$ . The paper establishes an upper bound of  $k^2 \log n$  on the number of bits used in a label. The best previous upper bound for the label size of such labeling scheme is  $2^k \log n$ .

**Key words:** Graph algorithms, vertex-connectivity, labeling schemes.

## 1 Introduction

### 1.1 Problem and Motivation

Network representations have played an extensive and often crucial role in many domains of computer science, ranging from data structures, graph algorithms to distributed computing and communication networks. Traditional network representations are usually global in nature; in order to derive useful information, one must access a global data structure representing the entire network, even if the sought information is local, pertaining to only a few nodes.

In contrast, the notion of *labeling schemes* (introduced in [6,7,16]) involves using a more *localized* representation of the network. The idea is to associate with each vertex a label, selected in a such way, that will allow us to infer information about any two vertices *directly* from their labels, without using *any* additional information sources. Hence in essence, this method bases the entire representation on the set of labels alone.

Obviously, labels of unrestricted size can be used to encode any desired information, including in particular the entire graph structure. Our focus is thus on informative labeling schemes using relatively *short* labels (say, of length polylogarithmic in  $n$ ). Labeling schemes of this type were recently developed for different graph families and for a variety of information types, including vertex adjacency [6,7,16,5,26], distance [27,20,15,14,12,17,29,8,1,26], tree routing [10,11,30], vertex-connectivity [21,5], flow [23,21], tree ancestry [3,4,19], nearest common ancestor in trees [2] and various other tree functions, such as center and separation level [28]. See [13] for a survey on static labeling schemes. The dynamic version was studied in [25,24,9,22].

---

\* Supported in part at the Technion by an Aly Kaufman fellowship.

The current paper studies informative labeling schemes supporting the vertex connectivity function of general graphs. This type of information may be useful in the decision making process required for various reservation-based routing and connection establishment mechanisms in communication networks, in which it is desirable to have accurate information about the potential number of available routes between any two given endpoints. We establish a labeling scheme supporting the  $k$ -vertex connectivity function of general  $n$ -node graphs using  $k^2 \log n$ -bit labels. The best previous upper bound for the label size of such a labeling scheme was shown in [21] to be  $2^k \log n$ .

## 1.2 Labeling Schemes

Let  $f$  be a function on pairs of vertices. An  $f$ -labeling scheme  $\pi = \langle \mathcal{M}, \mathcal{D} \rangle$  for the graph family  $\mathcal{G}$  is composed of the following components:

1. A *marker* algorithm  $\mathcal{M}$  that given a graph in  $\mathcal{G}$ , assigns labels to its vertices.
2. A polynomial time *decoder* algorithm  $\mathcal{D}$  that given the labels  $L(u)$  and  $L(v)$  of two vertices  $u$  and  $v$  in some graph in  $\mathcal{G}$ , outputs  $f(u, v)$ .

It is important to note that the decoder  $\mathcal{D}$ , responsible of the  $f$ -computation, is independent of  $G$  or of the number of vertices in it. Thus  $\mathcal{D}$  can be viewed as a method for computing  $f$ -values in a “distributed” fashion, given any set of labels and knowing that the graph belongs to some specific family  $\mathcal{G}$ . In contrast, the labels contain some information that can be precomputed by considering the whole graph structure. Therefore, in a sense, the area of labeling schemes can be considered as intermediate between the sequential and distributed fields.

The most commonly complexity measure used to evaluate a labeling scheme  $\pi = \langle \mathcal{M}, \mathcal{D} \rangle$  is the *Label Size*,  $\mathcal{L}_{\mathcal{M}}(\mathcal{G})$ : the maximum number of bits in a label assigned by the marker algorithm  $\mathcal{M}$  to any vertex in any graph in  $\mathcal{G}$ . Finally, given a function  $f$  and a graph family  $\mathcal{G}$ , let

$$\mathcal{L}(f, \mathcal{G}) = \min\{\mathcal{L}_{\mathcal{M}}(\mathcal{G}) \mid \exists \mathcal{D}, \langle \mathcal{M}, \mathcal{D} \rangle \text{ is an } f \text{ labeling scheme for } \mathcal{G}\}.$$

## 1.3 Vertex Connectivity

Let  $G = \langle V, E \rangle$  be an unweighted undirected graph. A set of paths  $P$  connecting the vertices  $u$  and  $w$  in  $G$  is *vertex-disjoint* if each vertex except  $u$  and  $w$  appears in at most one path  $p \in P$ . The *vertex-connectivity*  $\mathbf{v}\text{-conn}(u, w)$  of two vertices  $u$  and  $w$  in  $G$  equals the cardinality of the largest set  $P$  of vertex-disjoint paths connecting them. By Menger’s theorem (cf. [31]), for non-adjacent  $u$  and  $w$ ,  $\mathbf{v}\text{-conn}(u, w)$  equals the minimum number of vertices in  $G \setminus \{u, w\}$  whose removal from  $G$  disconnects  $u$  from  $w$ . (When a vertex is removed, all its incident edges are removed as well.)

## 1.4 Related Work and Our Contribution

Labeling schemes supporting the  $k$ -vertex connectivity function on general  $n$ -node graphs were previously studied in [21]. The label sizes achieved therein are



$\log n$  for  $k = 1$ ,  $3 \log n$  for  $k = 2$ ,  $5 \log n$  for  $k = 3$  and  $2^k \log n$  for  $k > 3$ . Based on a counting argument, the authors also present a lower bound of  $\Omega(k \log \frac{n}{k^3})$  for the required label size of such a labeling scheme.

In [5] the authors establish an adjacency labeling scheme on  $n$ -node trees using  $\log n + O(\log^* n)$ -bit labels. Using this adjacency labeling scheme they show how to reduce the upper bounds on the label size of  $k$ -vertex connectivity labeling schemes to  $2 \log n + O(\log^* n)$  in the the case  $k = 2$  and to  $4 \log n + O(\log^* n)$  in the the case  $k = 3$ .

In this paper we establish a  $k$ -vertex connectivity labeling scheme on general  $n$ -node graphs using  $k^2 \log n$ -bit labels.

## 2 Preliminaries

In an undirected graph  $G$ , two vertices  $u$  and  $v$  are called  $k$ -connected if there exist at least  $k$  vertex-disjoint paths between them, i.e., if  $\mathbf{v}\text{-conn}(u, v) \geq k$ . Given a graph  $G = \langle V, E \rangle$  and two vertices  $u$  from  $v$  in  $V$ , a set  $S \subseteq V$  separates  $u$  from  $v$  if  $u$  and  $v$  are not connected in the vertex induced subgraph  $G \setminus S$ .

**Theorem 1.** [Menger] (cf. [31]) *In an undirected graph  $G$ , two non-adjacent vertices  $u$  and  $v$  are  $k$ -connected iff no set  $S \subset G \setminus \{u, v\}$  of  $k - 1$  vertices can separate  $u$  from  $v$  in  $G$ .*

The  $k$ -connectivity graph of  $G = \langle V, E \rangle$  is  $C_k(G) = \langle V, E' \rangle$ , where  $(u, v) \in E'$  iff  $u$  and  $v$  are  $k$ -connected in  $G$ . A graph  $G$  is closed under  $k$ -connectivity if any two  $k$ -connected vertices in  $G$  are also neighbors in  $G$ . Let  $\mathcal{C}(k)$  be the family of all graphs which are closed under  $k$ -connectivity. Two subgraphs  $H, F \in \mathcal{C}(k)$  are vertex-disjoint subgraphs if they share no common vertex. For graphs  $G = \langle V, E \rangle$  and  $G_i = \langle V_i, E_i \rangle$ ,  $i > 1$ , we say that  $G$  can be decomposed into the  $G_i$ 's if  $\bigcup_i V_i = V$  and  $\bigcup_i E_i = E$ .

**Observation 2.** 1. *If  $G \in \mathcal{C}(k)$  then each connected component of  $G$  belongs to  $\mathcal{C}(k)$ .*  
2. *If  $G = H \cup F$  where  $H, F \in \mathcal{C}(k)$  are vertex-disjoint subgraphs of  $G$ , then  $G \in \mathcal{C}(k)$ .*

The following two lemmas are taken from [21].

**Lemma 1.** *For every graph  $G$ , if  $u$  and  $v$  are  $k$ -connected in  $C_k(G)$  then they are neighbors in  $C_k(G)$ , i.e.,  $C_k(G) \in \mathcal{C}(k)$ .*

**Lemma 2.** *Let  $G' = \langle V, E' \rangle$  where  $E' = E \cup \{(u, v)\}$  for some pair of  $k$ -connected vertices  $u$  and  $v$ . Then  $G$  and  $G'$  have the same  $k$ -connectivity graph, i.e.,  $C_k(G) = C_k(G')$ .*

A graph  $G$  is called  $k$ -orientable if there exists an orientation of the edges such that the out-degree of each vertex is bounded from above by  $k$ . The class of  $k$ -orientable graphs is denoted  $\mathcal{J}_{or}(k)$ .

**Observation 3.** *If  $G = H \cup F$  where  $H, F \in \mathcal{J}_{or}(k)$  are vertex-disjoint subgraphs of  $G$ , then  $G \in \mathcal{J}_{or}(k)$ .*

### 3 Vertex-Connectivity Labeling Schemes for General Graphs

Let  $\mathcal{G}_n$  denote the family of all undirected graphs with at most  $n$  vertices. In this section we present a  $k$ -vertex connectivity labeling scheme for the graph family  $\mathcal{G}_n$  using  $k^2 \log n$ -bit labels. Let  $\mathcal{C}_n(k) = \mathcal{C}(k) \cap \mathcal{G}_n$ , i.e.,  $\mathcal{C}_n(k)$  is the family of all graphs with at most  $n$  vertices, which are closed under  $k$ -connectivity.

As in [21], we rely on the basic observation that labeling  $k$ -connectivity for some graph  $G$  is equivalent to labeling adjacency for  $C_k(G)$ . By Lemma 1,  $C_k(G) \in \mathcal{C}_n(k)$  for every graph  $G \in \mathcal{G}_n$ . Therefore, instead of presenting a  $k$ -connectivity labeling scheme  $\mathcal{G}_n$ , we present an adjacency labeling scheme for the graph family  $\mathcal{C}_n(k)$ .

The general idea used in [21] for labeling adjacency for some  $G \in \mathcal{C}_n(k)$ , is to decompose  $G$  into a ‘simple’  $k$ -orientable graph in  $\mathcal{G}_n$  and two other graphs belonging to  $\mathcal{C}_n(k-1)$ . The labeling algorithm of [21] recursively labels subgraphs of  $G$  that belong to  $\mathcal{C}_n(t)$  for  $t < k$ . Since adjacency in a  $k$ -orientable graph in  $\mathcal{G}_n$  can be encoded using  $k \log n$ -bit labels, the resulted labels in the recursive labeling use at most  $2^k \log n$  bits. Our main technical contribution in this paper is to show that any graph  $G \in \mathcal{C}_n(k)$  can be decomposed into a ‘simple’  $k$ -orientable graph and (only) one other graph belonging to  $\mathcal{C}_n(k-1)$ . Thus, using a recursive labeling we obtain our  $k^2 \log n$  upper bound.

#### 3.1 The Decomposition

Consider a graph  $G \in \mathcal{C}_n(k)$ , and let  $C_1, \dots, C_m$  be its connected components. Fix  $i$  and let  $C = C_i$  be one of these connected components. By the first part in Observation 2,  $C \in \mathcal{C}(k)$ .

Let  $T \equiv T(C)$  denote a BFS tree spanning  $C$  rooted at some vertex  $r$ . For a vertex  $v \in C$ , let  $height(v)$  denote its height in  $T$ , i.e., its (hop) distance in  $T$  to the root  $r$ . An *uncle* of a vertex  $v \in C$  is a neighbor of  $v$  (in the graph  $C$ ) at height  $height(v) - 1$ . For every vertex  $v \in C$ , let  $Deg_{up}(v)$  denote the number of  $v$ 's uncles. For a vertex  $v \in C$ , let  $Deg_{up}^*(v) = \min(Deg_{up}(v), k)$ .

We now define a  $k$ -orientable subgraph of  $C$  called  $K$ . For each vertex  $v \in C$ , let  $U(v)$  be some set containing  $Deg_{up}^*(v)$  uncles of  $v$ . The graph  $K$  is the graph obtained by the set of edges  $\{(v, u) \mid v \in C \text{ and } u \in U(v)\}$ . Clearly,  $K$  is a  $k$ -orientable graph. Let  $H$  be the graph obtained from  $C$  by removing the edges of  $K$ , i.e.,  $H = C \setminus K$ . Note that  $H$  may not be connected.

**Lemma 3.** *If  $u$  and  $v$  are  $k-1$  connected in  $H$  then they are neighbors in  $C$ .*

*Proof.* Let  $u$  and  $v$  be two vertices which are  $k-1$  connected in  $H$ . Assume, by contradiction, that  $u$  and  $v$  are not neighbors in  $C$ . Since  $C \in \mathcal{C}(k)$ , then  $u$  and  $v$  are also not  $k$ -connected in  $C$ . Therefore, by Menger's theorem, since  $u$  and  $v$  are non-adjacent in  $C$ , there exist a set  $S \subset C \setminus \{u, v\}$  of  $k-1$  vertices which separates  $u$  from  $v$  in  $C$ . In particular,  $S$  separates  $u$  from  $v$  in  $H$ . Let  $s \in S$  be a vertex of lowest height among the vertices in  $S$ , and let  $S' = S \setminus \{s\}$ . Note

that since  $u$  and  $v$  are  $k - 1$  connected in  $H$ , there is no strict subset of  $S$  which separates  $u$  from  $v$  in  $H$ . In particular, there exists a path  $P$  in  $H \setminus \{S'\}$  which connects  $u$  with  $v$ . Note that  $s$  must belong to  $P$ .

We now show that in  $C$ , the set  $S$  does not separate  $u$  from the root  $r$  of  $C$ . If  $s$  is not an ancestor of  $u$  in  $T$  then clearly,  $S$  does not separate  $u$  from  $r$  even in  $T$ . Otherwise, if  $s$  is an ancestor of  $u$  in  $T$ , let  $w$  be the child of  $s$  in the path  $P$ . Note that  $w$  is also an ancestor of  $u$  (possibly  $w$  is  $u$  itself). Since the edge  $(w, s)$  belongs to  $H$ , it follows that  $w$  has at least  $k + 1$  uncles in  $C$ . Therefore,  $w$  has an uncle not in  $S$ . Moreover, since there is no vertex in  $S$  of smaller height than  $s$ , we obtain that there is a path connecting  $w$  and  $r$  in  $C \setminus \{S\}$ . Since the subpath of  $P$  connecting  $u$  and  $w$  also belongs to  $C \setminus \{S\}$ , we obtain that in  $C$ , the set  $S$  does not separate  $u$  from  $r$ . Similarly,  $v$  is connected to  $r$  in  $C \setminus \{S\}$ . It therefore follows that in  $C$ , the set  $S$  does not separate  $u$  from  $v$ . Contradiction. ■

**Lemma 4.**  *$C$  can be decomposed into a graph in  $\mathcal{C}(k-1)$  and a  $(k-1)$ -orientable graph.*

*Proof.* Transform the graph  $H$  into  $\widehat{H} = H \cup C_{k-1}(H)$  by adding the edges of  $C_{k-1}(H)$  to  $H$ , one by one. By induction on the steps of this process using Lemma 2, we get  $C_{k-1}(\widehat{H}) = C_{k-1}(H) \subseteq \widehat{H}$ . Therefore,  $\widehat{H} \in \mathcal{C}(k - 1)$ . By Lemma 3,  $\widehat{H}$  is a subgraph of  $C$ . It therefore follows that  $C$  can be decomposed to the graph  $\widehat{H} \in \mathcal{C}(k - 1)$  and the  $k$ -orientable graph  $K$ .

Using Observations 2 and 3, we obtain the following corollary.

**Corollary 1.** *Each  $G \in \mathcal{C}_n(k)$  can be decomposed into a graph in  $\mathcal{C}_n(k - 1)$  and a  $(k - 1)$ -orientable graph in  $\mathcal{G}_n$ .*

Using induction, we get the following corollary.

**Corollary 2.** *Each  $G \in \mathcal{C}_n(k)$  can be decomposed into a graph  $G_1 \in \mathcal{C}_n(1)$  and  $k - 1$  graphs  $G_2, G_3, \dots, G_k$  such that for each  $2 \leq i \leq k$ , the graph  $G_i$  is an  $(i - 1)$ -orientable graph in  $\mathcal{G}_n$ .*

### 3.2 The Labeling Scheme

We begin with the following simple observation.

**Observation 4.** *Let  $\mathcal{J}_n(k) = \mathcal{J}_{or}(k) \cap \mathcal{G}_n$  be the family of  $k$ -orientable graphs with at most  $n$  vertices. Then  $\mathcal{L}(\text{adjacency}, \mathcal{J}_n(k)) \leq (k + 1)\lceil \log n \rceil$ .*

*Proof.* Given a graph  $K \in \mathcal{J}_n(k)$ , we first assign a unique identifier  $id(u)$  in the range  $\{1, 2, \dots, n\}$  to each node  $u \in K$ . We may assume that  $id(u)$  is encoded using precisely  $\lceil \log n \rceil$  bits. (If  $id(u)$  is encoded using less than  $\lceil \log n \rceil$  bits, simply pad enough zeros to the left of the encoding.) We now orient the edges of the  $k$ -orientable graph  $K$  such that each node  $u \in K$ , points to at most  $k$  nodes.

For each node  $u \in K$ , let  $N(u)$  denote the set of nodes pointed by  $u$ . We have  $|N(u)| \leq k$ .

The label  $L_{or}(u)$  assigned to each node  $u$  consists of  $|N(u)|+1$  fields. Each field contains exactly  $\lceil \log n \rceil$  bits. The first field contains  $id(u)$ , and the other fields contain the identifiers of the nodes in  $N(u)$ . Given the labels  $L_{or}(u)$  and  $L_{or}(v)$  of two vertices  $u$  and  $v$ , the decoder outputs 1 iff either the first field in  $L_{or}(u)$  appears as a field in  $L_{or}(v)$  or the first field in  $L_{or}(v)$  appears as a field in  $L_{or}(u)$ . Clearly, this is a correct adjacency labeling scheme for  $\mathcal{J}_n(k)$  with label size at most  $(k+1)\lceil \log n \rceil$ . ■

**Theorem 5.**  $\mathcal{L}(\text{adjacency}, \mathcal{C}_n(k)) \leq k^2 \log n$ .

*Proof.* We describe an adjacency labeling scheme  $\pi = \langle \mathcal{M}, \mathcal{D} \rangle$  for  $\mathcal{C}_n(k)$ . Given a graph  $G \in \mathcal{C}_n(k)$ , we decompose  $G$  according to Corollary 2. Note that since a graph in  $\mathcal{C}_n(1)$  is simply a collection of cliques,  $\mathcal{L}(\text{adjacency}, \mathcal{C}_n(1)) \leq \lceil \log n \rceil$ . In other words, there exists an adjacency labeling scheme  $\pi_1 = \langle \mathcal{M}_1, \mathcal{D}_1 \rangle$  for  $\mathcal{C}_n(1)$  of size  $\lceil \log n \rceil$ . For a vertex  $u \in G_1$ , let  $L_1(u)$  denote the label given to  $u$  by  $\pi_1$ . By Observation 4, there exists an adjacency labeling scheme  $\pi_i = \langle \mathcal{M}_i, \mathcal{D}_i \rangle$  for  $\mathcal{J}_n(i)$  with label size  $(i+1)\lceil \log n \rceil$ . For each  $2 \leq i \leq k$  and each vertex  $u \in G_i$ , let  $L_i(u)$  denote the label given to  $u$  by  $\pi_{i-1}$ . As before, we may assume that for each  $1 \leq i \leq k$  and each vertex  $u \in G$ , the label  $L_i(u)$  is encoded using precisely  $i\lceil \log n \rceil$  bits.

For every vertex  $u \in G$ , the label given by the marker algorithm  $\mathcal{M}$  is  $L(u) = \langle L_1(u), L_2(u), \dots, L_k(u) \rangle$ . Given the labels  $L(u)$  and  $L(v)$  of two vertices  $u$  and  $v$  in some  $G \in \mathcal{C}(k)$ , the decoder  $\mathcal{D}$  outputs 1 iff there exists  $1 \leq i \leq k$  such that  $\mathcal{D}_i(L_i(u), L_i(v)) = 1$ .

The fact that the labeling scheme  $\pi$  is a correct adjacency labeling scheme for  $\mathcal{C}_n(k)$  follows from Corollary 2 and from the correctness of the adjacency labeling schemes  $\pi_i$ . The fact that the label size of  $\pi$  is at most  $\frac{k(k+1)}{2} \cdot \lceil \log n \rceil$  follows from Observation 4. If  $k > 3$ , then  $\frac{k(k+1)}{2} \cdot \lceil \log n \rceil \leq k^2 \log n$ . Therefore, the theorem follows by combining this inequality with the results of [21] for the cases  $k = 1, 2$  and  $3$ . ■

**Corollary 3.**  $\mathcal{L}(k - v\text{-conn}, \mathcal{G}_n) \leq k^2 \log n$ .

## References

1. Alstrup, S., Bille, P., Rauhe, T.: Labeling schemes for small distances in trees. In: Proc. 14th ACM-SIAM Symp. on Discrete Algorithms, ACM Press, New York (2003)
2. Alstrup, S., Gavoille, C., Kaplan, H., Rauhe, T.: Nearest Common Ancestors: A Survey and a new Distributed Algorithm. Theory of Computing Systems 37, 441–456 (2004)
3. Abiteboul, S., Kaplan, H., Milo, T.: Compact labeling schemes for ancestor queries. In: Proc. 12th ACM-SIAM Symp. on Discrete Algorithms, ACM Press, New York (2001)

4. Alstrup, S., Rauhe, T.: Improved Labeling Scheme for Ancestor Queries. In: Proc. 19th ACM-SIAM Symp. on Discrete Algorithms, ACM Press, New York (2002)
5. Alstrup, S., Rauhe, T.: Small induced-universal graphs and compact implicit graph representations. In: Proc. 43'rd annual IEEE Symp. on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos (2002)
6. Breuer, M.A.: Coding the vertexes of a graph. *IEEE Trans. on Information Theory* IT-12, 148–153 (1966)
7. Breuer, M.A., Folkman, J.: An unexpected result on coding the vertices of a graph. *J. of Mathematical Analysis and Applications* 20, 583–600 (1967)
8. Cohen, E., Halperin, E., Kaplan, H., Zwick, U.: Reachability and Distance Queries via 2-hop Labels. In: Proc. 13th ACM-SIAM Symp. on Discrete Algorithms, ACM Press, New York (2002)
9. Cohen, E., Kaplan, H., Milo, T.: Labeling dynamic XML trees. In: Proc. 21st ACM Symp. on Principles of Database Systems (June 2002)
10. Fraigniaud, P., Gavoille, C.: Routing in trees. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) *ICALP 2001*. LNCS, vol. 2076, pp. 757–772. Springer, Heidelberg (2001)
11. Fraigniaud, P., Gavoille, C.: A space lower bound for routing in trees. In: Proc. 19th Symp. on Theoretical Aspects of Computer Science (March 2002)
12. Gavoille, C., Paul, C.: Split decomposition and distance labelling: an optimal scheme for distance hereditary graphs. In: Proc. European Conf. on Combinatorics, Graph Theory and Applications (September 2001)
13. Gavoille, C., Peleg, D.: Compact and Localized Distributed Data Structures. *J. of Distributed Computing* 16, 111–120 (2003)
14. Gavoille, C., Katz, M., Katz, N.A., Paul, C., Peleg, D.: Approximate Distance Labeling Schemes. In: Meyer auf der Heide, F. (ed.) *ESA 2001*. LNCS, vol. 2161, pp. 476–488. Springer, Heidelberg (2001)
15. Gavoille, C., Peleg, D., Pérennes, S., Raz, R.: Distance labeling in graphs. In: Proc. 12th ACM-SIAM Symp. on Discrete Algorithms, pp. 210–219. ACM Press, New York (2001)
16. Kannan, S., Naor, M., Rudich, S.: Implicit representation of graphs. *SIAM J. on Discrete Math* 5, 596–603 (1992)
17. Kaplan, H., Milo, T.: Short and simple labels for small distances and other functions. In: Workshop on Algorithms and Data Structures (August 2001)
18. Kaplan, H., Milo, T.: Parent and ancestor queries using a compact index. In: Proc. 20th ACM Symp. on Principles of Database Systems (May 2001)
19. Kaplan, H., Milo, T., Shabo, R.: A Comparison of Labeling Schemes for Ancestor Queries. In: Proc. 19th ACM-SIAM Symp. on Discrete Algorithms, ACM Press, New York (2002)
20. Katz, M., Katz, N.A., Peleg, D.: Distance labeling schemes for well-separated graph classes. In: Proc. 17th Symp. on Theoretical Aspects of Computer Science, pp. 516–528 (February 2000)
21. Katz, M., Katz, N.A., Korman, A., Peleg, D.: Labeling schemes for flow and connectivity. *SIAM Journal on Computing* 34, 23–40 (2004)
22. Korman, A.: General Compact Labeling Schemes for Dynamic Trees. In: Proc. 19th International Symp. on Distributed Computing (September 2005)
23. Korman, A., Kutten, S.: Distributed Verification of Minimum Spanning Trees. Proc. 25th Annual Symposium on Principles of Distributed Computing (July 2006)
24. Korman, A., Peleg, D.: Labeling Schemes for Weighted Dynamic Trees. In: Proc. 30th Int. Colloq. on Automata, Languages & Prog. Eindhoven, The Netherlands, July 2003, SV LNCS (2003)

25. Korman, A., Peleg, D., Rodeh, Y.: Labeling schemes for dynamic tree networks. *Theory of Computing Systems* 37, 49–75 (2004)
26. Korman, A., Peleg, D., Rodeh, Y.: Constructing Labeling Schemes through Universal Matrices. In: Asano, T. (ed.) *ISAAC 2006*. LNCS, vol. 4288, Springer, Heidelberg (2006)
27. Peleg, D.: Proximity-preserving labeling schemes and their applications. In: *Proc. 25th Int. Workshop on Graph-Theoretic Concepts in Computer Science*, pp. 30–41 (1999)
28. Peleg, D.: Informative labeling schemes for graphs. In: Nielsen, M., Rovan, B. (eds.) *MFCS 2000*. LNCS, vol. 1893, pp. 579–588. Springer, Heidelberg (2000)
29. Thorup, M.: Compact oracles for reachability and approximate distances in planar digraphs. *J. of the ACM* 51, 993–1024 (2004)
30. Thorup, M., Zwick, U.: Compact routing schemes. In: *Proc. 13th ACM Symp. on Parallel Algorithms and Architecture*, pp. 1–10, Hersonissos, Crete, Greece, July (2001)
31. Even, S.: *Graph Algorithms*. Computer Science Press (1979)

# Unbounded-Error One-Way Classical and Quantum Communication Complexity

Kazuo Iwama<sup>1,\*</sup>, Harumichi Nishimura<sup>2,\*\*</sup>,  
Rudy Raymond<sup>3</sup>, and Shigeru Yamashita<sup>4,\*\*\*</sup>

<sup>1</sup> School of Informatics, Kyoto University, Kyoto 606-8501, Sakyo-ku,  
Yoshida-Honmachi, Japan  
`iwama@kuis.kyoto-u.ac.jp`

<sup>2</sup> School of Science, Osaka Prefecture University, Sakai 599-8531, Gakuen-cho, Japan  
`hnishimura@mi.s.osakafu-u.ac.jp`

<sup>3</sup> Tokyo Research Laboratory, IBM Japan, Yamato 242-8502,  
Simotsuruma 1623-14, Japan  
`raymond@jp.ibm.com`

<sup>4</sup> Graduate School of Information Science, Nara Institute of Science and Technology  
Nara 630-0192, Ikoma, Takayama-cho 8916-5, Japan  
`ger@is.naist.ac.jp`

**Abstract.** This paper studies the gap between quantum one-way communication complexity  $Q(f)$  and its classical counterpart  $C(f)$ , under the *unbounded-error* setting, i.e., it is enough that the success probability is strictly greater than  $1/2$ . It is proved that for *any* (total or partial) Boolean function  $f$ ,  $Q(f) = \lceil C(f)/2 \rceil$ , i.e., the former is always exactly one half as large as the latter. The result has an application to obtaining an exact bound for the existence of  $(m, n, p)$ -QRAC which is the  $n$ -qubit random access coding that can recover any one of  $m$  original bits with success probability  $\geq p$ . We can prove that  $(m, n, > 1/2)$ -QRAC exists if and only if  $m \leq 2^{2n} - 1$ . Previously, only the non-existence of  $(2^{2n}, n, > 1/2)$ -QRAC was known.

## 1 Introduction

*Communication complexity* is probably the most popular model for studying the performance gap between classical and quantum computations. Even if restricted to the one-way private-coin setting (which means no shared randomness or entanglement), several interesting developments have been reported in the last couple of years. For *promise problems*, i.e., if we are allowed to use the fact that inputs to Alice and Bob satisfy some special property, exponential gaps are known: Bar-Yossef, Jayram and Kerenidis [5] constructed a relation to provide

---

\* Supported in part by Scientific Research Grant, Ministry of Japan, 16092101.

\*\* Supported in part by Scientific Research Grant, Ministry of Japan, 19700011.

\*\*\* Supported in part by Scientific Research Grant, Ministry of Japan, 16092218 and 19700010.

an exponential gap,  $\Theta(\log n)$  vs.  $\Theta(\sqrt{n})$ , between one-way quantum and classical communication complexities. Recently, Gavinsky et al. [11] showed that a similar exponential gap also exists for a *partial* Boolean function.

For *total* Boolean functions, i.e., if there is no available promise, there are no known exponential or even non-linear gaps: As mentioned in [1], the equality function is a total Boolean function for which the one-way quantum communication complexity is approximately one half,  $(1/2 + o(1)) \log n$  vs.  $(1 - o(1)) \log n$ , of the classical counterpart. This is the largest known gap so far. On the other hand, there are total Boolean functions for which virtually no gap exists between quantum and classical communication complexities. For example, with  $p$  denoting the success probability, those complexity gaps are only a smaller order additive term,  $(1 - H(p))n$  vs.  $(1 - H(p))n + O(\log n)$ , for the index function [4,20], and  $n - 2 \log \frac{1}{2p-1}$  [21] vs.  $n$  (with  $n - O(\log \frac{1}{2p-1})$  as the lower bound) [18] for the inner product function. Note that all the results so far mentioned are obtained under the *bounded-error* assumption, i.e.,  $p \geq 1/2 + \alpha$  for some constant  $\alpha$  which is independent of the size of Boolean functions.

Thus there seem to be a lot of varieties, depending on specific Boolean functions, in the quantum/classical gap of one-way communication complexity. In this paper it is shown that such varieties completely disappear if we use the *unbounded-error* model where it is enough that the success probability is strictly greater than  $1/2$ .

## 1.1 Our Contribution

We show that one-way quantum communication complexity of any (total or partial) Boolean function is always exactly (without a deviation of even  $\pm 1$ ) one half of the one-way classical communication complexity in the *unbounded-error* setting. The study of unbounded-error (classical) communication complexity was initiated by Paturi and Simon [23]. They characterized almost tightly the unbounded-error one-way communication complexity of Boolean function  $f$ , denoted by  $C(f)$ , in terms of a geometrical measure  $k_f$  which is the minimum dimension of the arrangement of points and hyperplanes. Namely, they proved that  $\lceil \log k_f \rceil \leq C(f) \leq \lceil \log k_f \rceil + 1$ . We show that such a characterization is also applicable to the unbounded-error one-way *quantum* communication complexity  $Q(f)$ . To this end, we need to link accurately the one-way quantum communication protocol to the arrangement of points and hyperplanes, which turns out to be possible using geometric facts on quantum states [14,17]. As a result we show that  $Q(f) = \lceil \log(k_f + 1)/2 \rceil$ . Moreover, we also remove the small gap in [23], proving  $C(f) = \lceil \log(k_f + 1) \rceil$ . This enables us to provide the exact relation between  $Q(f)$  and  $C(f)$ , i.e.,  $Q(f) = \lceil C(f)/2 \rceil$ .

Our characterizations of  $Q(f)$  and  $C(f)$  have an application to quantum random access coding (QRAC) and classical random access coding (RAC) introduced by Ambainis et al. [4]. The  $(m, n, p)$ -QRAC (resp.  $(m, n, p)$ -RAC) is the  $n$ -qubit (resp.  $n$ -bit) coding that can recover any one of  $m$  bits with success probability  $\geq p$ . The asymptotic relation among the three parameters  $m, n, p$  was shown in [4] and [20]: If  $(m, n, p)$ -QRAC exists, then  $n \geq (1 - H(p))m$ ,



while there exists  $(m, n, p)$ -RAC if  $n \leq (1 - H(p))m + O(\log m)$ . This relation gives us a tight bound on  $n$  when  $p$  is relatively far from  $1/2$ . Unfortunately these inequalities give us little information under the unbounded-error setting or when  $p$  is very close to  $1/2$ , because the value of  $(1 - H(p))m$  become less than one. Hayashi et al. [13] showed that  $(m, n, p)$ -QRAC with  $p > 1/2$  does not exist when  $m = 2^{2n}$ . Our characterization directly shows that this is tight, that is,  $(m, n, > 1/2)$ -QRAC exists if and only if  $m \leq 2^{2n} - 1$ , which solves the remained open problem in [13]. A similar tight result on the existence of  $(m, n, > 1/2)$ -RAC is also obtained from our characterization. Moreover, we also give concrete constructions of such QRAC and RAC with an analysis of their success probability.

## 1.2 Related Work

We mainly focus on the gap between classical and quantum communication complexities.

**Partial/Total Boolean Functions.** For total functions, the one-way quantum communication complexity is nicely characterized or bounded below in several ways. Klauck [15] characterized the one-way communication complexity of total Boolean functions by the number of different rows of the communication matrix in the *exact* setting, i.e., the success probability is one, and showed that it equals to the one-way deterministic communication complexity. Also, he gave a lower bound of bounded-error one-way quantum communication complexity of total Boolean functions by the VC dimension. Aaronson [12] presented lower bounds of the one-way quantum communication complexity that are also applicable for partial Boolean functions. His lower bounds are given in terms of the deterministic or bounded-error classical communication complexity and the length of Bob's input, which are shown to be tight by using the partial Boolean function of Gavinsky et al. [11].

**One-way/Two-way/SMP Models.** Two-way communication model is also popular. It is known that the two-way communication complexity has a non-linear quantum/classical gap for total functions in the bounded-error model. The current largest gap is quadratic. Buhrman, Cleve and Wigderson [6] showed that the almost quadratic gap,  $O(\sqrt{n} \log n)$  vs.  $\Omega(n)$ , exists for the disjointness function. This gap was improved to  $O(\sqrt{n})$  vs.  $\Omega(n)$  in [3], which turned out to be optimal within a constant factor for the disjointness function [24]. On the contrary, in the unbounded-error setting, two-way communication model can be simulated by one-way model with only at most one bit additional communication [23]. In the simultaneous message passing (SMP) model where we have a referee other than Alice and Bob, an exponential quantum/classical gap for total functions was shown by Buhrman et al. [7].

**Private-coin/Public-coin Models.** The exponential quantum/classical separations in [5] and [11] still hold under the public-coin model where Alice and Bob share random coins, since the one-way classical public-coin model can be

simulated by the one-way classical private-coin model with additional  $O(\log n)$ -bit communication [19]. However, exponential quantum/classical separation for total functions remains open for all of the bounded-error two-way, one-way and SMP models. Note that the public-coin model is too powerful in the unbounded-error model: we can easily see that the unbounded-error one-way (classical or quantum) communication complexity of any function (or relation) is at most 1 with prior shared randomness.

**Unbounded-error Models.** Since the seminal paper [23], the unbounded-error (classical) one-way communication complexity has been developed in the literature [8,9,10]. Klauck [16] also studied a variant of the unbounded-error quantum and classical communication complexity, called the weakly unbounded-error communication complexity: the cost is communication (qu)bits plus  $\log 1/\epsilon$  where  $1/2 + \epsilon$  is the success probability. He characterized the discrepancy, a useful measure for bounded-error communication complexity [18], in terms of the weakly unbounded-error communication complexity.

## 2 Preliminaries

For basic notations of quantum computing, see [22]. In this paper, a “function” represents both total and partial Boolean functions.

**Communication Complexity.** The two-party communication complexity model is defined as follows. One party, say Alice, has input  $x$  from a finite set  $X$  and another party, say Bob, input  $y$  from a finite set  $Y$ . One of them, say, Bob wants to compute the value  $f(x, y)$  for a function  $f$ . (In some cases, relations are considered instead of functions.) Their communication process is called a *quantum (resp. classical) protocol* if the communication is done by using quantum bits (resp. classical bits). In particular, the protocol is called *one-way* if the communication is only from Alice to Bob. The communication cost of the protocol is the maximum number of (qu)bits needed over all  $(x, y) \in X \times Y$  by the protocol. The *unbounded-error one-way quantum (resp. classical) communication complexity* of  $f$ , denoted by  $Q(f)$  (resp.  $C(f)$ ), is the communication cost of the best one-way quantum (resp. classical) protocol with success probability strictly larger than  $1/2$ . In what follows, the term “classical” is often omitted when it is clear from the context. We denote the communication matrix of  $f$  by  $\mathbf{M}_f = ((-1)^{f(x,y)})$ . (We use the bold font letters for denoting vectors and matrices.)

**Arrangements.** The notion of arrangement has often been used as one of the basic concepts in computer science such as computational geometry and learning theory. The arrangement of points and hyperplanes has two well-studied measures: the minimum dimension and margin complexity. We use the former, as in [23], to characterize the unbounded-error one-way communication complexity (while the latter was used in [12] to give a lower bound of bounded-error quantum communication complexity under prior shared entanglement). A point in  $\mathbb{R}^n$  is denoted by the corresponding  $n$ -dimensional real vector. Also, a hyperplane  $\{(a_i) \in \mathbb{R}^n \mid \sum_{i=1}^n a_i h_i = h_{n+1}\}$  on  $\mathbb{R}^n$  is denoted by the  $(n + 1)$ -dimensional

real vector  $\mathbf{h} = (h_1, \dots, h_n, h_{n+1})$ , meaning that any point  $(a_i)$  on the plane satisfies the equation  $\sum_{i=1}^n a_i h_i = h_{n+1}$ . A  $\{1, -1\}$ -valued matrix  $\mathbf{M}$  on  $X \times Y$  is *realizable by an arrangement* of a set of  $|X|$  points  $\mathbf{p}_x = (p_1^x, \dots, p_k^x)$  and a set of  $|Y|$  hyperplanes  $\mathbf{h}_y = (h_1^y, \dots, h_k^y, h_{k+1}^y)$  in  $\mathbb{R}^k$  if for any  $x \in X$  and  $y \in Y$ ,  $\delta(\mathbf{p}_x, \mathbf{h}_y) := \text{sign}(\sum_{i=1}^k p_i^x h_i^y - h_{k+1}^y)$  is equal to  $\mathbf{M}(x, y)$ . Here,  $\text{sign}(a) = 1$  if  $a > 0$ ,  $-1$  if  $a < 0$ , and  $0$  otherwise. Intuitively, the point lies above, below, or on the plane if  $\delta(\mathbf{p}_x, \mathbf{h}_y) = 1, -1$ , and  $0$ , respectively. The value  $k$  is called the dimension of the arrangement. Let  $k_{\mathbf{M}}$  denote the smallest dimension of all arrangements that realize  $\mathbf{M}$ . In particular, if  $\mathbf{M} = \mathbf{M}_f$  then we say that  $f$  is *realized* by the arrangement, and denote  $k_{\mathbf{M}}$  by  $k_f$ .

**Bloch Vector Representations of Quantum States.** Mathematically, the  $N$ -level quantum state is represented by an  $N \times N$  positive matrix  $\rho$  satisfying  $\text{Tr}(\rho) = 1$ . (Note that if  $N = 2^n$  then  $\rho$  is considered as a quantum state that consists of  $n$  qubits.) In this paper we use  $N \times N$  matrices  $\mathbf{I}_N, \lambda_1, \dots, \lambda_{N^2-1}$ , called *generator matrices*, as a basis to represent  $N$ -level quantum states. Here,  $\mathbf{I}_N$  is the identity matrix (the subscript  $N$  is often omitted), and  $\lambda_i$ 's are the generators of  $SU(N)$  satisfying (i)  $\lambda_i = \lambda_i^\dagger$ , (ii)  $\text{Tr}(\lambda_i) = 0$  and (iii)  $\text{Tr}(\lambda_i \lambda_j) = 2\delta_{ij}$ . Then, the following lemma is known (see, e.g., [17]).

**Lemma 1.** *For any  $N$ -level quantum state  $\rho$  and any  $N \times N$  generator matrices  $\lambda_i$ 's, there exists an  $(N^2 - 1)$ -dimensional vector  $\mathbf{r} = (r_i)$  such that  $\rho$  can be written as*

$$\rho = \frac{1}{N} \left( \mathbf{I} + \sqrt{\frac{N(N-1)}{2}} \sum_{i=1}^{N^2-1} r_i \lambda_i \right). \tag{1}$$

The vector  $\mathbf{r}$  in this lemma is often called the *Bloch vector* of  $\rho$ . Note that  $\lambda_i$  can be any generator matrices satisfying the above conditions. In particular, it is well-known [22] that for  $N = 2$  one can choose  $\sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ ,  $\sigma_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ , and  $\sigma_3 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$  of Pauli matrices as  $\lambda_1, \lambda_2$ , and  $\lambda_3$ , respectively. Generally for  $N = 2^n$ , one can choose the tensor products of Pauli matrices (including  $\mathbf{I}$ ) multiplied by  $\sqrt{2/N}$  for  $\lambda_1, \dots, \lambda_{N^2-1}$ .

Note that Lemma 1 is a necessary condition for  $\rho$  to be a quantum state. Although our knowledge of the sufficient condition is relatively weak (say, see [14][17]), the following two lemmas on the mathematical description of  $N$ -level quantum states are enough for our purpose.

**Lemma 2** ([17]). *Let  $r = \sqrt{\sum_{i=1}^{N^2-1} r_i^2}$ . Then,  $\rho = \frac{1}{N} \left( \mathbf{I} + \sqrt{\frac{N(N-1)}{2}} \sum_{i=1}^{N^2-1} r_i \lambda_i \right)$  is a quantum state if and only if  $r \leq \sqrt{\frac{2}{N(N-1)} \frac{1}{|m(\sum_{i=1}^{N^2-1} (\frac{r_i}{r}) \lambda_i)|}}$ , where  $m(\mathbf{A})$  denotes the minimum of eigenvalues of a matrix  $\mathbf{A}$ , and  $\lambda_i$ 's are any generator matrices.*

**Lemma 3** ([14]). *Let  $B(\mathbb{R}^{N^2-1})$  be the set of Bloch vectors of all  $N$ -level quantum states. Let  $D_{r_s}(\mathbb{R}^{N^2-1}) = \{\mathbf{r} \in \mathbb{R}^{N^2-1} \mid |\mathbf{r}| \leq \frac{1}{N-1}\}$  (called the small*

ball), and  $D_{r_i}(\mathbb{R}^{N^2-1}) = \{\mathbf{r} \in \mathbb{R}^{N^2-1} \mid |\mathbf{r}| \leq 1\}$  (called the large ball). Then,  $D_{r_s}(\mathbb{R}^{N^2-1}) \subseteq B(\mathbb{R}^{N^2-1}) \subseteq D_{r_i}(\mathbb{R}^{N^2-1})$ .

### 3 Quantum Tight Bound

In [13], we gave a geometric view of the quantum protocol on random access coding. It turns out that this view together with the notion of arrangements is a powerful tool for characterizing the unbounded-error one-way quantum communication complexity.

**Theorem 1.**  $Q(f) = \lceil \log(k_f + 1)/2 \rceil$  for every function  $f : X \times Y \rightarrow \{0, 1\}$ .

The outline of the proof is as follows: In Lemma 4 we first establish a relation similar to Lemma 1 between a POVM (Positive Operator-Valued Measure)  $\{\mathbf{E}, \mathbf{I} - \mathbf{E}\}$  over  $n$  qubits and a  $(2^{2n} - 1)$ -dimensional (Bloch) vector  $\mathbf{h}(\mathbf{E})$ . Then, we prepare Lemma 5 to show that the measurement results of POVM  $\{\mathbf{E}, \mathbf{I} - \mathbf{E}\}$  on a state  $\rho$  correspond to the arrangement operation  $\delta(\mathbf{r}(\rho), \mathbf{h}(\mathbf{E}))$ , where  $\mathbf{r}(\rho)$  is the Bloch vector for  $\rho$ .

Now in order to prove  $Q(f) \geq \lceil \log(k_f + 1)/2 \rceil$ , suppose that there is a protocol whose communication complexity is  $n$ . This means for any  $x \in X$  and  $y \in Y$ , we have  $n$ -qubit states  $\rho_x$  and POVMs  $\{\mathbf{E}_y, \mathbf{I} - \mathbf{E}_y\}$  such that: (i) the dimensions of  $\mathbf{r}(\rho_x)$  and  $\mathbf{h}(\mathbf{E}_y)$  are  $2^{2n} - 1$  and  $2^{2n}$  (by Lemmas 1 and 4, and note that  $N = 2^n$ ), and (ii)  $\mathbf{M}_f(x, y) = \text{sign}(\text{Tr}(\mathbf{E}_y \rho_x) - 1/2) = \delta(\mathbf{r}(\rho_x), \mathbf{h}(\mathbf{E}_y))$  (the first equality by the assumption and the second one by Lemma 5). By (ii) we can conclude that the arrangement of points  $\mathbf{r}(\rho_x)$  and hyperplanes  $\mathbf{h}(\mathbf{E}_y)$  realizes  $f$ , and by (i) its dimension is  $2^{2n} - 1$ . Thus,  $k_f$  is at most  $2^{2n} - 1$ , implying that  $n (= Q(f)) \geq \lceil \log(k_f + 1)/2 \rceil$ .

To prove the converse, suppose that there exists an  $(N^2 - 1)$ -dimensional arrangement of points  $\mathbf{r}_x$  and hyperplanes  $\mathbf{h}_y$  realizing  $f$ . For simplicity, suppose that  $N^2 - 1 = k_f$  (see the proof of Theorem 1 for the details). Let us fix some generator matrices  $\lambda_i$ 's. However,  $\rho_x$  obtained directly from  $\lambda_i$ 's and  $\mathbf{r}_x$  by Eq. (1) may not be a valid quantum state. Fortunately, by Lemma 6 we can simply multiply  $\mathbf{r}_x$  by a fixed constant factor to obtain  $\mathbf{r}'_x$  such that  $\mathbf{r}'_x$  lies in the small ball in Lemma 3 and therefore corresponds to an  $n$ -qubit state  $\rho(\mathbf{r}'_x)$ . Similarly, by Lemma 7 we can get  $\mathbf{h}'_y$  corresponding to POVM  $\{\mathbf{E}(\mathbf{h}'_y), \mathbf{I} - \mathbf{E}(\mathbf{h}'_y)\}$ . Obviously, the arrangement of points  $\mathbf{r}'_x$  and hyperplanes  $\mathbf{h}'_y$  realizes  $f$ , its dimension is the same  $N^2 - 1$  and the corresponding  $\rho(\mathbf{r}'_x)$  and  $\{\mathbf{E}(\mathbf{h}'_y), \mathbf{I} - \mathbf{E}(\mathbf{h}'_y)\}$  are an  $N$ -level (or  $\lceil \log N \rceil$ -qubit) quantum state and a POVM over  $N$ -level quantum states, respectively. Now, by Lemma 5, we can compute  $f(x, y)$  by  $\text{sign}(\text{Tr}(\mathbf{E}(\mathbf{h}'_y) \rho(\mathbf{r}'_x)) - 1/2)$ , which means  $Q(f) \leq \lceil \log N \rceil = \lceil \log(k_f + 1)/2 \rceil$ .

According to the above outline, we start to present technical lemmas whose proofs are omitted. The following lemma, shown similarly as Lemma 1, is a necessary condition for  $\{\mathbf{E}, \mathbf{I} - \mathbf{E}\}$  to be a POVM.

**Lemma 4.** For any POVM  $\{\mathbf{E}, \mathbf{I} - \mathbf{E}\}$  over  $N$ -level quantum states and  $N \times N$  generator matrices  $\lambda_i$ 's, there exists an  $N^2$ -dimensional vector  $\mathbf{e} = (e_i)$  such that  $\mathbf{E}$  can be written as  $\mathbf{E} = e_{N^2} \mathbf{I} + \sum_{i=1}^{N^2-1} e_i \lambda_i$ .

We call the above vector  $\mathbf{e} = (e_1, \dots, e_{N^2-1}, e_{N^2})$  the *Bloch vector* of POVM  $\{\mathbf{E}, \mathbf{I} - \mathbf{E}\}$ . The next lemma relates the probability distribution of binary values obtained by measuring a quantum state  $\rho$  with a POVM  $\{\mathbf{E}, \mathbf{I} - \mathbf{E}\}$  with their Bloch vectors.

**Lemma 5.** *Let  $\mathbf{r} = (r_i) \in \mathbb{R}^{N^2-1}$  and  $\mathbf{e} = (e_i) \in \mathbb{R}^{N^2}$  be the Bloch vectors of an  $N$ -level quantum state  $\rho$  and a POVM  $\{\mathbf{E}, \mathbf{I} - \mathbf{E}\}$ . Then, the probability that the measurement value 0 is obtained is*

$$\text{Tr}(\mathbf{E}\rho) = e_{N^2} + \sqrt{\frac{2(N-1)}{N}} \sum_{i=1}^{N^2-1} r_i e_i.$$

The last two lemmas provide a shrink-and-shift mapping from any real vectors and hyperplanes to, respectively, Bloch vectors of quantum states lying in the small ball of Lemma 3 and POVMs.

**Lemma 6.** (1) *For any  $\mathbf{r} = (r_1, r_2, \dots, r_k) \in \mathbb{R}^k$  and  $N$  satisfying  $N^2 \geq k + 1$ ,*

$$\rho(\mathbf{r}) = \frac{1}{N} \left( \mathbf{I} + \sqrt{\frac{N(N-1)}{2}} \sum_{i=1}^k \left( \frac{r_i}{|\mathbf{r}|(N-1)} \right) \lambda_i \right)$$

*is an  $N$ -level quantum state.*

(2) *If  $\rho(\mathbf{r})$  is a quantum state, then  $\rho(\gamma\mathbf{r})$  is also a quantum state for any  $\gamma \leq 1$ .*

**Lemma 7.** *For any hyperplane  $\mathbf{h} = (h_1, \dots, h_k, h_{k+1}) \in \mathbb{R}^{k+1}$ , let  $N$  be any number such that  $N^2 \geq k + 1$ , and let  $\alpha, \beta$  be two positive numbers that are at most  $\frac{1}{2(|h_{k+1}| + h\sqrt{\frac{2(N-1)}{N}})}$  where  $h = \sum_{i=1}^k h_i^2$ . Then, the  $N^2$ -dimensional vector*

*defined by  $\mathbf{h}(\alpha, \beta) = (\beta h_1, \dots, \beta h_k, 0, \dots, 0, 1/2 - \alpha h_{k+1})$  is the Bloch vector of a POVM  $\{\mathbf{E}_0, \mathbf{E}_1\}$  over  $N$ -level quantum states, where  $\mathbf{E}_0$  and  $\mathbf{E}_1$  are given as*

$$\mathbf{E}_0 = \left( \frac{1}{2} - \alpha h_{k+1} \right) \mathbf{I} + \beta \sum_{i=1}^k h_i \lambda_i \quad \text{and} \quad \mathbf{E}_1 = \left( \frac{1}{2} + \alpha h_{k+1} \right) \mathbf{I} - \beta \sum_{i=1}^k h_i \lambda_i. \quad (2)$$

Now we prove our main theorem in this section.

**Proof of Theorem 1.**  $k_f$  is simply written as  $k$  in this proof.

$(Q(f) \geq \lceil \log(k+1)/2 \rceil)$ . Let  $n = Q(f)$  and  $N = 2^n$ . Assume that there is an  $n$ -qubit protocol for  $f$ . That is, Alice on input  $x$  sends an  $n$ -qubit state  $\rho_x$  to Bob with input  $y$ . He then measures  $\rho_x$  with a POVM  $\{\mathbf{E}_y, \mathbf{I} - \mathbf{E}_y\}$  so that  $\text{sign}(\text{Tr}(\mathbf{E}_y \rho_x) - 1/2) = \mathbf{M}_f(x, y)$ . From Lemmas 1 and 4 we can define the points  $\mathbf{p}_x = (p_i^x) \in \mathbb{R}^{N^2-1}$  and hyperplanes  $\mathbf{h}_y = (h_i^y) \in \mathbb{R}^{N^2}$  so that  $\mathbf{p}_x$  is the Bloch vector of  $\rho_x$ , and  $\mathbf{h}_y = \left( \sqrt{\frac{2(N-1)}{N}} e_1^y, \dots, \sqrt{\frac{2(N-1)}{N}} e_{N^2-1}^y, 1/2 - e_{N^2}^y \right)$  where  $\mathbf{e}_y = (e_i^y)$  is the Bloch vector of the POVM  $\{\mathbf{E}_y, \mathbf{I} - \mathbf{E}_y\}$ . Notice that by

Lemma 5.  $\text{Tr}(\mathbf{E}_y \rho_x) = e_{N^2}^y + \sqrt{\frac{2(N-1)}{N}} \sum_{i=1}^{N^2-1} p_i^x e_i^y$ , which is  $> 1/2$  if  $\mathbf{M}_f(x, y) = 1$  and  $< 1/2$  if  $\mathbf{M}_f(x, y) = -1$  by assumption. Thus, we can see that

$$\delta(\mathbf{p}_x, \mathbf{h}_y) = \text{sign} \left( e_{N^2}^y + \sqrt{\frac{2(N-1)}{N}} \sum_{i=1}^{N^2-1} p_i^x e_i^y - 1/2 \right) = \mathbf{M}_f(x, y),$$

meaning that there exists an arrangement of points and hyperplanes in  $\mathbb{R}^{N^2-1}$  which realizes  $f$ . Thus, by definition,  $k$  is at most  $N^2 - 1 = 2^{2n} - 1$  which implies  $Q(f) = n \geq \lceil \log(k+1)/2 \rceil$ .

( $Q(f) \leq \lceil \log(k+1)/2 \rceil$ ). Suppose that there is a  $k$ -dimensional arrangement of points  $\mathbf{p}_x = (p_i^x) \in \mathbb{R}^k$  and hyperplanes  $\mathbf{h}_y = (h_i^y) \in \mathbb{R}^{k+1}$  that realizes  $\mathbf{M}_f$ . That is,  $\delta(\mathbf{p}_x, \mathbf{h}_y) = \mathbf{M}_f(x, y)$  for every  $(x, y) \in X \times Y$ . By carefully shrinking and shifting this arrangement into Bloch vectors in the small ball, we will show the construction of an  $n$ -qubit protocol for  $f$ , that is,  $n$ -qubit states  $\rho_x$  for Alice and POVMs  $\{\mathbf{E}_y, \mathbf{I} - \mathbf{E}_y\}$  for Bob with the smallest  $n$  satisfying  $k \leq 2^{2n} - 1$ , and hence obtain  $Q(f) \leq n = \lceil \log(k+1)/2 \rceil$ .

Let  $\gamma_x = \min \left\{ \frac{1}{|\mathbf{p}_x|(2^{2n}-1)}, \frac{1}{2^{2n}-1} \right\}$  for each  $x \in X$ . Then, since  $(2^n)^2 \geq k+1$ ,

Lemma 6 implies that  $\frac{1}{2^n} \left( \mathbf{I} + \sqrt{\frac{2^n(2^{2n}-1)}{2}} \sum_{i=1}^k \gamma_x p_i^x \boldsymbol{\lambda}_i \right)$  is an  $n$ -qubit state, and hence  $\gamma_x \mathbf{p}_x$  is the Bloch vector of its qubit state. Moreover, Lemma 7 implies that by taking  $\beta_y = \frac{1}{2 \left( |h_{k+1}^y| + \sqrt{\sum_{i=1}^k (h_i^y)^2} \sqrt{\frac{2(2^{2n}-1)}{2^n}} \right)}$ ,  $\mathbf{h}_y(\beta_y, \beta_y) = (\beta_y h_1^y, \dots, \beta_y h_k^y, 0, \dots, 0, 1/2 - \beta_y h_{k+1}^y)$  is the Bloch vector of a POVM over  $n$ -qubit states.

Now let  $\gamma = \frac{1}{\sqrt{2}} \min_{x \in X} \gamma_x$ ,  $\beta = \min_{y \in Y} \beta_y$ , and  $\alpha = \sqrt{\frac{2(2^{2n}-1)}{2^n}} \gamma \beta$ . Since  $\gamma \leq \gamma_x$  for any  $x \in X$  and  $0 < \alpha < \beta \leq \beta_y$  for any  $y \in Y$ , Lemmas 6(2) and 7 show that  $\gamma \mathbf{p}_x$  and  $\mathbf{h}_y(\beta, \alpha)$  are also the Bloch vectors of an  $n$ -qubit state  $\rho_x$  and a POVM  $\{\mathbf{E}_y, \mathbf{I} - \mathbf{E}_y\}$  over  $n$ -qubit states, respectively. By Lemma 5, the probability that the measurement value 0 is obtained is

$$\begin{aligned} \text{Tr}(\mathbf{E}_y \rho_x) &= \frac{1}{2} - \alpha h_{k+1}^y + \sqrt{\frac{2(2^{2n}-1)}{2^n}} \gamma \beta \sum_{i=1}^k p_i^x h_i^y = \frac{1}{2} + \alpha \left( \sum_{i=1}^k p_i^x h_i^y - h_{k+1}^y \right) \\ &= \begin{cases} > 1/2 & \text{if } \mathbf{M}_f(x, y) = 1 \\ < 1/2 & \text{if } \mathbf{M}_f(x, y) = -1, \end{cases} \end{aligned}$$

where the last inequality comes from the assumption. Therefore, the states  $\rho_x$  and POVMs  $\{\mathbf{E}_y, \mathbf{I} - \mathbf{E}_y\}$  can be used to obtain an  $n$ -qubit protocol for  $f$ .  $\square$

Combined with the results in [8,10], Theorem 1 gives us a nontrivial bound for the inner product function  $IP_n$  (i.e.,  $IP_n(x, y) = \sum_{i=1}^n x_i y_i \pmod{2}$  for any  $x = x_1 \cdots x_n \in \{0, 1\}^n$  and  $y = y_1 \cdots y_n \in \{0, 1\}^n$ ). Note that the bounded-error quantum communication complexity is at least  $n - O(1)$ , and  $n/2 - O(1)$  even if we allow two-way protocol and prior entanglement [21].

**Corollary 1.**  $\lceil n/4 \rceil \leq Q(IP_n) \leq \lceil ((\log 3)n + 2)/4 \rceil$ .

## 4 Classical Tight Bound

Paturi and Simon [23] shows that for every function  $f : X \times Y \rightarrow \{0, 1\}$ ,  $\lceil \log k_f \rceil \leq C(f) \leq \lfloor \log k_f \rfloor + 1$ . We remove this small gap as follows.

**Theorem 2.**  $C(f) = \lceil \log(k_f + 1) \rceil$  for every function  $f : X \times Y \rightarrow \{0, 1\}$ .

*Proof.* Let  $k = k_f$  in this proof.

( $C(f) \geq \lceil \log(k + 1) \rceil$ ). Let  $N = 2^{C(f)}$ . Suppose that there is a  $C(f)$ -bit protocol for  $f$ . Paturi and Simon (in Theorem 2 in [23]) gave an  $N$ -dimensional arrangement of points  $\mathbf{p}_x = (p_i^x) \in \mathbb{R}^N$  and hyperplanes  $\mathbf{h}_y = (h_1^y, \dots, h_N^y, 1/2) \in \mathbb{R}^{N+1}$ , that is,  $\delta(\mathbf{p}_x, \mathbf{h}_y) = \mathbf{M}_f(x, y)$  for every  $(x, y) \in X \times Y$ . Noting that the points  $\mathbf{p}_x$  are probabilistic vectors satisfying  $\sum_{i=1}^N p_i = 1$ , we can reduce the dimension of the arrangement to  $N - 1$ . We define  $\mathbf{q}_x = (q_i^x) \in \mathbb{R}^{N-1}$  and  $\mathbf{l}_y = (l_i^y) \in \mathbb{R}^N$  from  $\mathbf{p}_x$  and  $\mathbf{h}_y$ , respectively, as follows:  $\mathbf{q}_x = (p_1^x, p_2^x, \dots, p_{N-1}^x)$  and  $\mathbf{l}_y = (h_1^y - h_N^y, h_2^y - h_N^y, \dots, h_{N-1}^y - h_N^y, \frac{1}{2} - h_N^y)$ . From the assumption and  $p_N^x = 1 - \sum_{i=1}^{N-1} p_i^x$ ,

$$\begin{aligned} \sum_{i=1}^{N-1} q_i^x l_i^y - l_N^y &= \sum_{i=1}^{N-1} p_i^x (h_i^y - h_N^y) - \frac{1}{2} + h_N^y = \sum_{i=1}^{N-1} p_i^x h_i^y - \frac{1}{2} + h_N^y - \sum_{i=1}^{N-1} p_i^x h_N^y \\ &= \sum_{i=1}^N p_i^x h_i^y - \frac{1}{2} = \begin{cases} > 0 & \text{if } \mathbf{M}_f(x, y) = 1 \\ < 0 & \text{if } \mathbf{M}_f(x, y) = -1. \end{cases} \end{aligned}$$

Thus,  $\delta(\mathbf{q}_x, \mathbf{l}_y) = \mathbf{M}_f(x, y)$  for every  $(x, y) \in X \times Y$ . That is,  $M_f$  is realizable by the  $(N - 1)$ -dimensional arrangement of points  $\mathbf{q}_x$  and hyperplanes  $\mathbf{l}_y$ . By definition,  $k \leq N - 1 = 2^{C(f)} - 1$ , which means that  $C(f) \geq \lceil \log(k + 1) \rceil$ .

( $C(f) \leq \lceil \log(k + 1) \rceil$ ). The proof is also based on that of Theorem 2 of Paturi and Simon [23]. They showed the existence of a protocol where Alice (with input  $x$ ) sends a probabilistic mixture of (at most)  $k + 2$  different messages to Bob (with input  $y$ ). In this proof we reduce the number of messages to  $k + 1$ . That is, we construct the following protocol using  $k + 1$  different messages: Alice sends a message  $S_j$  with probability  $q_j^x$  where  $j \in [k + 1]$ , and Bob outputs 0 with probability  $l_j^y$  upon receiving  $S_j$ . Here,  $[n] := \{1, 2, \dots, n\}$  for any  $n \in \mathbb{N}$ . We will show that the probability that Bob outputs 0, represented as  $\sum_{j=1}^{k+1} q_j^x l_j^y$ , is  $> 1/2$  if  $\mathbf{M}_f(x, y) = 1$  and  $< 1/2$  if  $\mathbf{M}_f(x, y) = -1$ .

Assume that there exists a  $k$ -dimensional arrangement of points  $\mathbf{p}_x = (p_i^x) \in \mathbb{R}^k$  and hyperplanes  $\mathbf{h}_y = (h_i^y) \in \mathbb{R}^{k+1}$  that realizes  $\mathbf{M}_f$ , that is,  $\delta(\mathbf{p}_x, \mathbf{h}_y) = \mathbf{M}_f(x, y)$  for every  $(x, y) \in X \times Y$ . Let  $s = \max_{x \in X} \max_{i \in [k]} |p_i^x|$ ,  $\alpha_x = 1 + \sum_{i=1}^k (s + p_i^x)$  for each  $x \in X$ , and  $\beta_y = \max(|h_1^y|, \dots, |h_k^y|, |h_{k+1}^y| + s \sum_{i=1}^k |h_i^y|)$  for each  $y \in Y$ . Then, we define  $\mathbf{q}_x = (q_i^x) \in \mathbb{R}^{k+1}$  and  $\mathbf{l}_y = (l_i^y) \in \mathbb{R}^{k+1}$  by  $\mathbf{q}_x = \left( \frac{s+p_1^x}{\alpha_x}, \frac{s+p_2^x}{\alpha_x}, \dots, \frac{s+p_k^x}{\alpha_x}, \frac{1}{\alpha_x} \right)$  and  $\mathbf{l}_y = \left( \frac{1}{2} + \frac{h_1^y}{2\beta_y}, \frac{1}{2} + \frac{h_2^y}{2\beta_y}, \dots, \frac{1}{2} - \frac{h_{k+1}^y + s \sum_{i=1}^k h_i^y}{2\beta_y} \right)$ . It

can be easily checked that  $0 \leq q_i^x \leq 1$  for all  $(x, i) \in X \times [k]$ ,  $\sum_{i=1}^{k+1} q_i^x = 1$ , and  $0 \leq l_i^y \leq 1$  for all  $(y, i) \in Y \times [k+1]$ . Moreover,

$$\begin{aligned} \sum_{i=1}^{k+1} q_i^x l_i^y &= \sum_{i=1}^k \left( \frac{s + p_i^x}{\alpha_x} \right) \left( \frac{1}{2} + \frac{h_i^y}{2\beta_y} \right) + \frac{1}{\alpha_x} \left( \frac{1}{2} - \frac{h_{k+1}^y + s \sum_{i=1}^k h_i^y}{2\beta_y} \right) \\ &= \frac{1}{2} + \frac{1}{2\alpha_x\beta_y} \left( \sum_{i=1}^k h_i^y p_i^x - h_{k+1}^y \right) = \begin{cases} > 1/2 & \text{if } \mathbf{M}_f(x, y) = 1 \\ < 1/2 & \text{if } \mathbf{M}_f(x, y) = -1. \end{cases} \end{aligned}$$

Hence, given a  $k$ -dimensional arrangement of points and hyperplanes realizing  $\mathbf{M}_f$ , we can construct a protocol using at most  $k+1$  different messages for  $f$ . This means that  $C(f) \leq \lceil \log(k+1) \rceil$ . This completes the proof.

Now we obtain our main result in this paper.

**Theorem 3.** *For every function  $f : X \times Y \rightarrow \{0, 1\}$ ,  $Q(f) = \lceil C(f)/2 \rceil$ .*

## 5 Applications to Random Access Coding

In this section we discuss the random access coding as an application of our characterizations of  $Q(f)$  and  $C(f)$ . The concept of quantum random access coding (QRAC) and the classical random access coding (RAC) were introduced by Ambainis et al. [4]. The  $(m, n, p)$ -QRAC (resp.  $(m, n, p)$ -RAC) is an encoding of  $m$  bits using  $n$  qubits (resp.  $n$  bits) so that *any* one of the  $m$  bits can be obtained with probability at least  $p$ . In fact, the function computed by the RAC (or QRAC) is known before as the *index* function in the context of communication complexity. It is denoted as  $INDEX_n(x, i) = x_i$  for any  $x \in \{0, 1\}^n$  and  $i \in [n]$  (see [18]).

### 5.1 Existence of QRAC and RAC

First we use Theorems 1 and 2 to show the existence of RAC and QRAC. As seen in [23], the smallest dimension of arrangements realizing  $INDEX_n$  is  $n$ . Thus, Theorem 1 gives us the following corollary for its unbounded-error one-way quantum communication complexity.

**Corollary 2.**  $Q(INDEX_n) = \lceil \log(n+1)/2 \rceil$ .

Similarly, Theorem 2 gives its classical counterpart, which is tighter than [23].

**Corollary 3.**  $C(INDEX_n) = \lceil \log(n+1) \rceil$ .

Since random access coding is the same as  $INDEX_n$  as Boolean functions, the following tight results are obtained for the existence of random access coding.

**Corollary 4.**  $(2^{2n} - 1, n, > 1/2)$ -QRAC exists, but  $(2^{2n}, n, > 1/2)$ -QRAC does not exist. Moreover,  $(2^n - 1, n, > 1/2)$ -RAC exists, but  $(2^n, n, > 1/2)$ -RAC does not exist.

Corollary 4 solves the open problem in [13] in its best possible form. It also implies the non-existence of  $(2, 1, > 1/2)$ -RAC shown in [4]. Note that this fact does not come directly from the characterization of  $C(f)$  in [23].



## 5.2 Explicit Constructions of QRAC and RAC

In this subsection, we give an explicit construction of  $(2^{2n} - 1, n, > 1/2)$ -QRAC and  $(2^n - 1, n, > 1/2)$ -RAC that leads to a better success probability than what obtained from direct applications of Theorems [1](#) and [2](#). For the case of QRAC, the construction is based on the proof idea of Theorem [1](#) combined with the property of the index function. Their proofs are omitted due to space constraint.

**Theorem 4.** *For any  $n \geq 1$ , there exists a  $(2^{2n} - 1, n, p)$ -QRAC such that  $p \geq \frac{1}{2} + \frac{1}{2\sqrt{(2^n - 1)(2^{2n} - 1)}}$ .*

We can also obtain the upper bound of the success probability of  $(2^{2n} - 1, n, p)$ -QRAC from the asymptotic bound by Ambainis et al. [4](#): For any  $(2^{2n} - 1, n, p)$ -QRAC,  $p \leq \frac{1}{2} + \sqrt{\frac{(\ln 2)n}{2^{2n} - 1}}$ . It remains open to close the gap between the lower and upper bounds of the success probability.

Similarly, for the case of RAC we have the following theorem.

**Theorem 5.** *There exists a  $(2^n - 1, n, p)$ -RAC such that  $p \geq \frac{1}{2} + \frac{1}{2(2^{n+1} - 5)}$ .*

The success probability of  $(2^n - 1, n, p)$ -RAC can also be bounded by the asymptotic bound in [4](#): For any  $(2^n - 1, n, p)$ -RAC,  $p \leq \frac{1}{2} + \sqrt{\frac{(\ln 2)n}{2^n - 1}}$ .

## 6 Concluding Remarks

We proved the tight possible separation between unbounded-error one-way quantum and classical communication complexities for Boolean functions. However, it still remains open whether similar results hold for the two-way case, and for the case of computing non-Boolean functions (such as relational problems).

## Acknowledgements

R.R. would like to thank David Avis of McGill Univ. for introducing the world of arrangements, and Kazuyoshi Hidaka and Hiroyuki Okano of Tokyo Research Lab. of IBM Japan for their supports. We also thank Tsuyoshi Ito of Univ. of Tokyo and Hans Ulrich Simon of Ruhr-Universität Bochum for helpful discussion.

## References

1. Aaronson, S.: Limitation of quantum advice and one-way communication. *Theory of Computing* 1, 1–28 (2005)
2. Aaronson, S.: The learnability of quantum states, [quant-ph/0608142](#)
3. Aaronson, S., Ambainis, A.: Quantum search of spatial regions. *Theory of Computing* 1, 47–79 (2005)
4. Ambainis, A., Nayak, A., Ta-shma, A., Vazirani, U.: Dense quantum coding and a lower bound for 1-way quantum automata. In: *Proc. 31st STOC*, pp. 376–383 (1999) Journal version appeared in *J. ACM* 49, 496–511 (2002)

5. Bar-Yossef, Z., Jayram, T.S., Kerenidis, I.: Exponential separation of quantum and classical one-way communication complexity. Proc. 36th STOC, pp. 128–137 (2004)
6. Buhrman, H., Cleve, R., Wigderson, A.: Quantum vs. classical communication and computation. Proc. 30th STOC, pp. 63–68 (1998)
7. Buhrman, H., Cleve, R., Watrous, J., de Wolf, R.: Quantum fingerprinting. Phys. Rev. Lett. 87, Article no 167902 (2001)
8. Forster, J.: A linear lower bound on the unbounded error probabilistic communication complexity. J. Comput. Syst. Sci. 65, 612–625 (2002)
9. Forster, J., Krause, M., Lokam, S.V., Mubarakzjanov, R., Schmitt, N., Simon, H.U.: Relation between communication complexity, linear arrangements, and computational complexity. In: Hariharan, R., Mukund, M., Vinay, V. (eds.) FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science. LNCS, vol. 2245, pp. 171–182. Springer, Heidelberg (2001)
10. Forster, J., Simon, H.U.: On the smallest possible dimension and the largest possible margin of linear arrangements representing given concept classes. Theoret. Comput. Sci. 350, 40–48 (2006)
11. Gavinsky, D., Kempe, J., Kerenidis, I., Raz, R., de Wolf, R.: Exponential separations for one-way quantum communication complexity, with applications to cryptography. In: Proc. 39th STOC (to appear). Also, quant-ph/0611209
12. Gavinsky, D., Kempe, J., de Wolf, R.: Strengths and weaknesses of quantum fingerprinting. In: Proc. 21st CCC, pp. 288–298 (2006)
13. Hayashi, M., Iwama, K., Nishimura, H., Raymond, R., Yamashita, S. (4,1)-quantum random access coding does not exist – one qubit is not enough to recover one of four bits. New J. Phys. 8, Article no. 129 (2006)
14. Jakóbczyk, L., Siennicki, M.: Geometry of Bloch vectors in two-qubit system. Phys. Lett. A 286, 383–390 (2001)
15. Klauck, H.: On quantum and probabilistic communication: Las Vegas and one-way protocols. In: Proc. 32nd STOC, pp. 644–651 (2000)
16. Klauck, H.: Lower bounds for quantum communication complexity. In: Proc. 42nd FOCS, pp. 288–297 (2001)
17. Kimura, G., Kossakowski, A.: The Bloch-vector space for  $N$ -level systems – the spherical-coordinate point of view. Open Sys. Information Dyn. 12, 207–229 (2005). Also, quant-ph/0408014
18. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge (1997)
19. Newman, I.: Private vs. common random bits in communication complexity. Inform. Process. Lett. 39, 67–71 (1991)
20. Nayak, A.: Optimal lower bounds for quantum automata and random access codes. In: Proc. 40th IEEE FOCS, pp. 369–376 (1999)
21. Nayak, A., Salzman, J.: Limits on the ability of quantum states to convey classical messages. J. ACM 53, 184–206 (2006)
22. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information, Cambridge (2000)
23. Paturi, R., Simon, J.: Probabilistic communication complexity. J. Comput. Syst. Sci. 33, 106–123 (1986)
24. Razborov, A.: Quantum communication complexity of symmetric predicates. Izvestiya Mathematics 67, 145–159 (2003)

# A Lower Bound on Entanglement-Assisted Quantum Communication Complexity

Ashley Montanaro<sup>1</sup> and Andreas Winter<sup>2</sup>

<sup>1</sup> Department of Computer Science, University of Bristol, Bristol, BS8 1UB, U.K  
`montanar@cs.bris.ac.uk`

<sup>2</sup> Department of Mathematics, University of Bristol, Bristol, BS8 1TW, U.K  
`a.j.winter@bristol.ac.uk`

**Abstract.** We prove a general lower bound on the bounded-error entanglement-assisted quantum communication complexity of Boolean functions. The bound is based on the concept that any classical or quantum protocol to evaluate a function on distributed inputs can be turned into a quantum communication protocol. As an application of this bound, we give a very simple proof of the statement that almost all Boolean functions on  $n + n$  bits have communication complexity linear in  $n$ , even in the presence of unlimited entanglement.

## 1 Introduction

The field of communication complexity aims to characterise the information that is required to be transmitted between two or more separated parties in order for them to compute some function of their joint inputs. Following the field's inception by Yao [25], it has been found to have many links to other areas of computer science. Here we will be concerned with the generalisation to *quantum* communication complexity. (See [24] and [15] for excellent introductions to quantum and classical communication complexity, respectively.)

Specifically, consider a total Boolean function  $f : \{0, 1\}^n \times \{0, 1\}^n \mapsto \{0, 1\}$ . The quantum communication complexity of  $f$  is defined to be the minimum number of qubits required to be transmitted between two parties (Alice and Bob) for them to compute  $f(x, y)$  for any two  $n$ -bit inputs  $x, y$ , given that Alice starts out with  $x$  and Bob with  $y$ . This number is clearly upper-bounded by  $n$ , but for some functions may be considerably lower. Alice and Bob may be allowed some probability of error  $\epsilon$ , and may be allowed to share an entangled state before they start their protocol. We will assume that Bob has to output the result.

Some functions are known to have a quantum communication complexity lower than their classical communication complexity (for example, a bounded-error protocol for the disjointness function  $f(x, y) = 1 \Leftrightarrow |x \wedge y| = 0$  requires  $\Omega(n)$  bits of classical communication, but only  $\Theta(\sqrt{n})$  qubits of quantum communication [1, 22]), but the extent of the possible reduction in communication is unknown. In particular, it is still open whether the quantum communication

complexity of total functions can ever be exponentially smaller than the classical communication complexity, although it has been shown [21] that quantum communication can exponentially reduce the communication cost of computing a partial function (where there is a promise on the input). It is therefore of interest to produce lower bounds on quantum communication complexity. In this context, the model with prior entanglement is less well understood; although there are strong bounds known for some classes of functions [5,22], there are few general lower bounds [4].

In this paper, we develop an elegant result of Cleve et al. that relates computation to communication. Cleve et al. showed [5] that, if Alice and Bob have access to a protocol to exactly compute the inner product function  $IP(x, y) = \sum_i x_i y_i \pmod{2}$ , then this can be used to produce a quantum protocol that communicates Alice's input  $x$  to Bob. They used this to show that  $IP$  cannot be computed (exactly and without prior entanglement) by sending fewer than  $n$  bits from Alice to Bob. Similar results hold for the bounded-error case and with prior entanglement.

We show that a weaker form of this result can be extended to *all* Boolean functions. That is, for almost any Boolean function  $f$ , the ability for Alice and Bob to compute  $f$  implies the ability for Alice to send some arbitrary information to Bob. The extension leads to the development of a new complexity measure for Boolean functions: *communication capacity*. Given a Boolean function  $f(x, y)$ , we define the communication capacity of  $f$  as the maximum number of bits which the execution of a protocol to compute  $f$  allows Alice to communicate to Bob (in an asymptotic sense). As we will show, this gives a lower bound on the quantum communication complexity of  $f$ , with or without entanglement. It is easy to see that concept has no classical analogue: if Alice and Bob are given a black-box protocol to compute a Boolean function and are only allowed to input classical data to it, each use of the protocol can communicate at most one bit of arbitrary information.

Some comments on notation: we will use  $M$  to denote the square communication matrix of  $f$  (where  $M_{xy}$  is equal to  $(-1)^{f(x,y)}$ ).  $H(v)$  will denote the Shannon entropy of a vector  $v$  ( $H(v) = -\sum_i v_i \log v_i$ ), and  $S(\rho)$  the von Neumann entropy of a density matrix  $\rho$  ( $S(\rho) = -\text{tr}(\rho \log \rho)$ , i.e. the Shannon entropy of the eigenvalues of  $\rho$ ). All logarithms will be taken to base 2. We assume familiarity with quantum computation [20].

We will use the standard notation  $Q_E(f)$  to denote the quantum communication complexity of  $f$  in the case where the protocol must be exact,  $Q_\epsilon(f)$  the complexity where Alice and Bob are allowed to err with probability  $\epsilon < 1/2$ , and  $Q_2(f)$  the complexity in the case where  $\epsilon = 1/3$ . In all three cases, Alice and Bob's initial state is separable;  $Q_E^*(f)$ ,  $Q_\epsilon^*(f)$  and  $Q_2^*(f)$  will represent the equivalent quantities in the case where they are allowed to share an arbitrary initial entangled state.

Then our main result can be stated as follows (a more precise statement is given as Theorem 2 below).

**Theorem 1.** *Let  $f : \{0, 1\}^n \times \{0, 1\}^n \mapsto \{0, 1\}$  be a total Boolean function with communication matrix  $M$ . Then, for any diagonal matrices  $A$  and  $B$  where  $A_{ii}, B_{ii} \geq 0$  for all  $i$ , and where  $\sum_i A_{ii}^2 = \sum_i B_{ii}^2 = 1$ ,*

$$Q_2^*(f) = \Omega(H(\sigma^2(AMB))/\log n) \quad (1)$$

where  $\sigma^2(M)$  is the vector of squared singular values of a matrix  $M$ , i.e. the eigenvalues of  $M^\dagger M$ .

This lower bound is a generalisation of a bound obtained by Klauck [13] on quantum communication complexity in the model without entanglement (his bound is the special case where  $A_{ii} = B_{ii} = 1/\sqrt{2^n}$ ). The result here can thus be seen as extending Klauck’s bound to the model of entanglement-assisted quantum communication, and giving it a satisfying operational interpretation. As our bound also holds for classical communication complexity, it fits into the framework of results using ideas from quantum information to say something about classical computation.

As is usual in computational complexity, we would expect most functions to have “high” quantum communication complexity. Kremer showed [14] by a counting argument that a random function  $f$  has  $Q_2(f) \geq n/2$  (and thus  $Q_E(f) \geq n/2$ ). Buhrman and de Wolf extended Kremer’s methods to show that, for all  $f$ ,  $Q_E^*(f) \geq (\log \text{rank}(f))/2$  [4] (an equivalent result is shown in section 6.4.2 of [19]). As almost all Boolean matrices have full rank, this shows that for almost all  $f$ ,  $Q_E^*(f) \geq n/2$ . Very recently, Gavinsky, Kempe and de Wolf [8] have shown the final remaining case: for almost all  $f$ ,  $Q_2^*(f) = \Omega(n)$ . Their technique was to relate quantum communication protocols to quantum fingerprinting protocols, and then to show a relationship between quantum fingerprinting and some well-studied concepts from classical computational learning theory. Linial and Shraibman also obtained this result very recently [16], by extending the well-known discrepancy lower bound to entanglement-assisted quantum communication complexity.

As an application of our communication capacity technique, we reprove the result that for almost all  $f$ ,  $Q_2^*(f) = \Omega(n)$ . The proof is of a quite different character and of (arguably) a more “quantum” nature, as it is based on showing that the entropy of almost all density matrices produced in a certain random way is high.

Following the completion of this work, Linial and Shraibman have shown [17] that the minimum  $\gamma_2$  norm of matrices that approximate the communication matrix  $M$  gives a lower bound on entanglement-assisted quantum communication complexity. This norm is defined as

$$\gamma_2(M) = \min_{XY=M} \|X\|_{\ell_2 \rightarrow \ell_\infty} \|Y\|_{\ell_1 \rightarrow \ell_2} \quad (2)$$

where  $\|X\|_{\ell_2 \rightarrow \ell_\infty}$  is the largest  $\ell_2$  norm of a row of  $X$ , and  $\|Y\|_{\ell_1 \rightarrow \ell_2}$  is the largest  $\ell_2$  norm of a column of  $Y$ . Among other results, Linial and Shraibman use this lower bound to extend the bound of Klauck [13] to the model of quantum communication with entanglement. Their work thus proves the special case of Theorem 1 where  $A_{ii} = B_{ii} = 1/\sqrt{2^n}$ .

## 2 Turning Any Distributed Function into a Communication Protocol

In this section, we will describe how to change any protocol for evaluating a distributed function into a communication protocol. This is an extension of the protocol in [5] for the IP function, but for some functions, the communication will be considerably more inefficient than IP allows (Alice may only be able to send  $\ll n$  bits to Bob).

### 2.1 Exact Protocols

Say Alice and Bob have access to a classical or quantum protocol that computes  $f(x, y)$  exactly. We express this as a unitary  $P$  that performs the following action.

$$P|x\rangle_A|y\rangle_B|0\rangle_B|a\rangle_{AB} = |x\rangle_A|y\rangle_B|f(x, y)\rangle_B|a'\rangle_{AB} \tag{3}$$

where  $|a\rangle, |a'\rangle$  are arbitrary (and possibly entangled) ancilla states shared by Alice and Bob. Note that, as  $P$  does not modify the first two registers, we may decompose it as follows:

$$P = \sum_{x,y} |x\rangle\langle x|_A \otimes |y\rangle\langle y|_B \otimes U_{xy} \tag{4}$$

for some unitary  $U_{xy}$  acting only on the last two registers. Following [5], we will turn this into a “clean” protocol  $P'$  by giving Bob an additional qubit to copy the answer into, then running the protocol backwards to uncompute the “junk”  $|a'\rangle$ . The steps of the clean protocol are thus

- (i)  $\rightarrow |x\rangle_A|y\rangle_B|0\rangle_B|0\rangle_B|a\rangle_{AB}$
- (ii)  $\rightarrow |x\rangle_A|y\rangle_B|f(x, y)\rangle_B|0\rangle_B|a'\rangle_{AB}$
- (iii)  $\rightarrow |x\rangle_A|y\rangle_B|f(x, y)\rangle_B|f(x, y)\rangle_B|a'\rangle_{AB}$
- (iv)  $\rightarrow |x\rangle_A|y\rangle_B|0\rangle_B|f(x, y)\rangle_B|a\rangle_{AB}$

where now the fourth register contains the answer. Ignoring the third and fifth registers, which are the same at the beginning and the end of the protocol, we are left with the map

$$P'|x\rangle_A|y\rangle_B|0\rangle_B = |x\rangle_A|y\rangle_B|f(x, y)\rangle_B \tag{5}$$

Note that, if the original protocol  $P$  communicated  $a$  qubits from Alice to Bob and  $b$  qubits from Bob to Alice, the protocol  $P'$  requires  $a + b$  qubits to be communicated in each direction. That is,  $P'$  sends as many qubits in the “forward” direction as the original protocol  $P$  sends in total. Now say Alice wants to communicate her input  $x$  to Bob using this protocol. They start with the following state, where  $(b_y)$  is an arbitrary probability distribution on Bob’s inputs:

$$|\psi\rangle = |x\rangle_A \left( \sum_{y \in \{0,1\}^n} \sqrt{b_y} |y\rangle_B \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)_B \tag{6}$$

Note that this state is separable (so we do not *require* entanglement to execute the communication protocol). After executing the clean protocol for  $f$ , they are left with

$$P'|\psi\rangle = |x\rangle_A \left( \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}^n} \sqrt{b_y} |y\rangle_B (|f(x,y)\rangle - |1-f(x,y)\rangle)_B \right) \quad (7)$$

$$= |x\rangle_A \left( \sum_{y \in \{0,1\}^n} (-1)^{f(x,y)} \sqrt{b_y} |y\rangle_B \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)_B \quad (8)$$

Ignoring the registers that remain the same throughout, Bob has the following state at the end of the protocol.

$$|\psi_x\rangle = \sum_{y \in \{0,1\}^n} (-1)^{f(x,y)} \sqrt{b_y} |y\rangle \quad (9)$$

This state provides some information about Alice’s bit string  $x$ . If  $\langle \psi_x | \psi_{x'} \rangle = 0$  for all  $x' \neq x$  (as is the case with the protocol of [5] for IP, where Bob uses the uniform distribution on his inputs) then Bob can determine  $x$  with certainty and hence has received  $n$  bits from Alice. If this is not the case, then we can still quantify precisely how much information can be transmitted. The protocol is equivalent to Alice encoding the classical bit-string  $x$  as a state  $|\psi_x\rangle$ , and co-operating with Bob to send it to him. Say Alice uses a distribution  $(a_x)$  on her inputs. Then the ensemble representing what Bob eventually receives is

$$\rho = \sum_{x \in \{0,1\}^n} a_x |\psi_x\rangle \langle \psi_x| \quad (10)$$

By Holevo’s theorem [10], the entropy  $S(\rho)$  describes the maximum number of bits of classical information about  $x$  available to Bob by measuring  $\rho$ . And, by the Holevo-Schumacher-Westmoreland channel coding theorem for a channel with pure signal states [9], Alice and Bob can achieve this bound (in an asymptotic sense) using block coding!

Therefore, the ability to compute  $f$  exactly can be used to transmit  $S(\rho)$  bits of information through a quantum channel, even though this does not hold if Alice and Bob are restricted to a classical channel. We thus define the *communication capacity* of a Boolean function  $f$  as the maximum over all probability distributions  $(a_x)$  (on Alice’s inputs) and  $(b_y)$  (on Bob’s inputs) of

$$S \left( \sum_{x \in \{0,1\}^n} a_x |\psi_x\rangle \langle \psi_x| \right), \text{ where } |\psi_x\rangle = \sum_{y \in \{0,1\}^n} (-1)^{f(x,y)} \sqrt{b_y} |y\rangle \quad (11)$$

## 2.2 Bounded Error Protocols

In the case where Alice and Bob have access to a protocol computing  $f$  with some probability of error, Bob will not have the state  $|\psi_x\rangle$  at the end of the

protocol, but rather some approximation  $|\psi_x^\epsilon\rangle$ . We will now show that, if the error probability is small, this is in fact still sufficient to communicate a significant amount of information from Alice to Bob. As before, Alice will use a distribution  $(a_x)$  on her inputs, and Bob a distribution  $(b_y)$ .

Say Alice and Bob are using a protocol  $P^\epsilon$  that computes  $f$  with probability of error  $\epsilon$ , where  $\epsilon < 1/2$ . As before, the  $|x\rangle$  and  $|y\rangle$  registers will be unchanged by this protocol, so we can write

$$P^\epsilon = \sum_{x,y} |x\rangle\langle x|_A \otimes |y\rangle\langle y|_B \otimes U_{xy}^\epsilon \tag{12}$$

Now let us run the protocol on the same starting state  $|\psi\rangle$  as in the previous section.

$$\begin{aligned} \text{(i)} \quad & |x\rangle_A \left( \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}^n} \sqrt{b_y} |y\rangle_B |0\rangle_B (|0\rangle - |1\rangle)_B \right) |a\rangle_{AB} \\ \text{(ii)} \rightarrow & |x\rangle_A \left( \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}^n} \sqrt{b_y} |y\rangle_B (\alpha_{xy}|0\rangle + \beta_{xy}|1\rangle)_B (|0\rangle - |1\rangle)_B \right) |a'\rangle_{AB} \end{aligned}$$

where the effect of  $U_{xy}^\epsilon$  on the “answer” qubit has been decomposed into  $\alpha_{xy}$  and  $\beta_{xy}$  components. If  $f(x, y) = 0$ , then  $|\alpha_{xy}|^2 \geq 1 - \epsilon$ , and thus (by unitarity)  $|\beta_{xy}|^2 \leq \epsilon$ ; if  $f(x, y) = 1$ ,  $|\beta_{xy}|^2 \geq 1 - \epsilon$  and  $|\alpha_{xy}|^2 \leq \epsilon$ . The ancilla register is still completely arbitrary, and in particular may be entangled with any of the other registers. Continuing the protocol, we have

$$\begin{aligned} \text{(iii)} \rightarrow & |x\rangle_A \left( \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}^n} \sqrt{b_y} |y\rangle_B (\alpha_{xy}|0\rangle(|0\rangle - |1\rangle) - \beta_{xy}|1\rangle(|0\rangle - |1\rangle))_B \right) |a'\rangle_{AB} \\ \text{(iv)} \rightarrow & |x\rangle_A \left( \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}^n} \sqrt{b_y} |y\rangle_B (\alpha_{xy}(\alpha_{xy}^*|0\rangle + \gamma_{xy}^*|1\rangle)|0\rangle - \alpha_{xy}(\alpha_{xy}^*|0\rangle + \right. \\ & \left. + \gamma_{xy}^*|1\rangle)|1\rangle - \beta_{xy}(\beta_{xy}^*|0\rangle + \delta_{xy}^*|1\rangle)|0\rangle + \beta_{xy}(\beta_{xy}^*|0\rangle + \delta_{xy}^*|1\rangle)|1\rangle)_B \right) |a\rangle_{AB} \end{aligned}$$

where we introduce  $\gamma_{xy}^*$  and  $\delta_{xy}^*$  as arbitrary elements of  $(U_{xy}^\epsilon)^\dagger$ , subject only to the constraint that  $U_{xy}^\epsilon$  be unitary. We may now remove registers that end the protocol unchanged and rewrite Bob’s final state as

$$|\psi_x^\epsilon\rangle = \sum_{y \in \{0,1\}^n} \sqrt{b_y} |y\rangle \left( (|\alpha_{xy}|^2 - |\beta_{xy}|^2)|0\rangle + (\alpha_{xy}\gamma_{xy}^* - \beta_{xy}\delta_{xy}^*)|1\rangle \right) \tag{13}$$

Now, if  $f(x, y) = 0$ , then  $|\alpha_{xy}|^2 - |\beta_{xy}|^2 \geq 1 - 2\epsilon > 0$ , whereas if  $f(x, y) = 1$ ,  $|\alpha_{xy}|^2 - |\beta_{xy}|^2 \leq 2\epsilon - 1 < 0$ . We may therefore write

$$|\psi_x^\epsilon\rangle = \sum_{y \in \{0,1\}^n} \sqrt{b_y} |y\rangle \left( (-1)^{f(x,y)} \cos \theta_{xy} |0\rangle + e^{i\phi_{xy}} \sin \theta_{xy} |1\rangle \right) \tag{14}$$



where  $\theta_{xy}$  is real with  $\cos \theta_{xy} \geq 1 - 2\epsilon$ , and  $\phi_{xy}$  is an arbitrary phase. Crucially, the form of these states is quite restricted and close to the original  $|\psi_x\rangle$ . In fact, it is clear that

$$|\langle \psi_x | \langle 0 | \rangle | \psi_x^\epsilon \rangle|^2 \geq (1 - 2\epsilon)^2 \tag{15}$$

Set  $\rho^\epsilon = \sum_{x \in \{0,1\}^n} a_x |\psi_x^\epsilon\rangle \langle \psi_x^\epsilon|$  (this is what Bob receives). We will compare this to the state  $\rho' = \sum_{x \in \{0,1\}^n} a_x |\psi_x\rangle |0\rangle \langle \psi_x| \langle 0|$ , where of course  $S(\rho') = S(\rho)$ . We have

$$\|\rho' - \rho^\epsilon\|_1 \leq 2\sqrt{1 - (1 - 2\epsilon)^2} \leq 4\sqrt{\epsilon} \tag{16}$$

We will use Fannes' inequality [6] to show that  $S(\rho^\epsilon) \approx S(\rho)$ . Define the function

$$\eta_0(x) = \begin{cases} -x \log x & \text{for } x \leq 1/e \\ 1/e \log e & \text{for } x > 1/e \end{cases} \tag{17}$$

Then Fannes' inequality gives that

$$S(\rho^\epsilon) \geq S(\rho) - 4\sqrt{\epsilon}n - \log \eta_0(4\sqrt{\epsilon}) \tag{18}$$

and we are done.

### 2.3 Communication Complexity Lower Bounds from Communication Capacity

A lower bound for the communication capacity of a function  $f$  can be written down in terms of its communication matrix  $M$  as follows. As before, set

$$\rho = \sum_{x \in \{0,1\}^n} a_x |\psi_x\rangle \langle \psi_x| \text{ for } |\psi_x\rangle = \sum_{y \in \{0,1\}^n} (-1)^{f(x,y)} \sqrt{b_y} |y\rangle \tag{19}$$

for arbitrary probability distributions  $(a_x), (b_y)$  on Alice and Bob's inputs. Define the rescaled Gram matrix  $G$  as  $G_{ij} = \sqrt{a_i} \sqrt{a_j} \langle \psi_i | \psi_j \rangle$ . Now it is known [12] that  $G$  will have the same eigenvalues as  $\rho$ , and thus the same entropy. But it can easily be verified that

$$G = (AMB)(AMB)^\dagger \tag{20}$$

where  $A$  and  $B$  are diagonal matrices with  $A_{ii} = \sqrt{a_i}, B_{ii} = \sqrt{b_i}$ . So the eigenvalues of  $G$  are simply the singular values squared of  $AMB$ . We may thus write

$$S(\rho) = H(\sigma^2(AMB)) \tag{21}$$

where  $\sigma^2(M)$  denotes the vector containing the squared singular values of a matrix  $M$ , i.e. the eigenvalues of  $M^\dagger M$ . We can now produce lower bounds on the quantum communication complexity of  $f$  by appealing to the result of Nayak and Salzman [18] which states that, if Alice wishes to transmit  $n$  bits to Bob over a quantum channel with probability of success  $p$ , Alice must send  $m \geq \frac{1}{2} \left( n - \log \frac{1}{p} \right)$  bits to Bob. If they are not allowed to share prior entanglement, the factor of  $1/2$  vanishes. This immediately gives a lower bound on the

exact quantum communication complexity of  $f$ , as lower bounds on the forward communication required for the “clean” protocols that we use translate into lower bounds on the total amount of communication needed for any communication protocol.

In the bounded-error case, we can still use the Nayak-Salzman result. Consider a block coding scheme with blocks of length  $k$  where each letter  $|\psi_x^\epsilon\rangle$  is produced by one use of  $f$ , as in the previous section. By [9] there exists such a scheme that transmits  $kS(\rho^\epsilon) - o(k)$  bits of information with  $k$  uses of  $f$ , as  $k \rightarrow \infty$ , and probability of success  $p \rightarrow 1$ . A lower bound on the bounded-error quantum communication complexity of  $f$  follows immediately:

$$mk \geq \frac{1}{2}(kS(\rho^\epsilon) - o(k) - o(1)), \tag{22}$$

hence, after taking the limit  $k \rightarrow \infty, p \rightarrow 1$ , we find  $m \geq \frac{1}{2}S(\rho^\epsilon)$ .

In order to reduce the error probability  $\epsilon$  to  $O(1/n^2)$  (to remove the additive term linear in  $n$  in inequality (18)), it is sufficient to repeat the original protocol  $O(\log n)$  times and take a majority vote [14]. Alternatively, using (18) directly gives a better bound for functions for which  $S(\rho)$  is linear in  $n$ . We thus have the following theorem.

**Theorem 2.** *Let  $f : \{0, 1\}^n \times \{0, 1\}^n \mapsto \{0, 1\}$  be a total Boolean function with communication matrix  $M$ . Then, for any diagonal matrices  $A$  and  $B$  where  $A_{ii}, B_{ii} \geq 0$  for all  $i$ , and where  $\sum_i A_{ii}^2 = \sum_i B_{ii}^2 = 1$ ,*

$$Q_E(f) \geq H(\sigma^2(AMB)) \tag{23}$$

$$Q_E^*(f) \geq \frac{1}{2}H(\sigma^2(AMB)) \tag{24}$$

$$Q_\epsilon(f) \geq \begin{cases} \Omega(H(\sigma^2(AMB))/\log n) \\ H(\sigma^2(AMB)) - 4\sqrt{\epsilon}n - \log \eta_0(4\sqrt{\epsilon}) \end{cases} \tag{25}$$

$$Q_\epsilon^*(f) \geq \begin{cases} \Omega(H(\sigma^2(AMB))/\log n) \\ \frac{1}{2}(H(\sigma^2(AMB)) - 4\sqrt{\epsilon}n - \log \eta_0(4\sqrt{\epsilon})) \end{cases} \tag{26}$$

where  $\eta_0(x)$  is defined as in [17].

If we use the uniform distribution on Alice and Bob’s inputs, then  $AMB = M/2^n$ . In the case of the models without entanglement, Klauck obtained this specialised result via a different method [13]. This theorem can thus be seen as simultaneously extending Klauck’s work to the model with entanglement, generalising it, and giving it an operational interpretation. The special case of the uniform distribution was also used by Cleve et al. [5] to prove their lower bound on the communication complexity of IP.

Note that, as  $H(\sigma^2(AMB))$  is upper bounded by  $\log \text{rank}(M)$ , the previously known “log rank” lower bound [4] is superior for lower bounding exact quantum communication complexity ( $Q_E(f)$  and  $Q_E^*(f)$ ).

### 3 Rényi Entropic Bounds on Communication Capacity

A disadvantage of the von Neumann entropy  $S(\rho)$  is the difficulty involved in its computation. The *second Rényi entropy*  $S_2(\rho)$  [23] provides an easily computable lower bound on  $S(\rho)$ .  $S_2(\rho)$  is defined as

$$S_2(\rho) = -\log \operatorname{tr}(\rho^2) = -\log \sum_{i,j} |\rho_{ij}|^2 \quad (27)$$

and we have the fundamental property that  $S_2(\rho) \leq S(\rho)$ . The Rényi entropy also obeys the bounds  $0 \leq S_2(\rho) \leq n$ . As with the von Neumann entropy, the Rényi entropy is a function only of the eigenvalues of  $\rho$ , so the Rényi entropy of the density matrix corresponding to an ensemble of equiprobable states is the same as that of the rescaled Gram matrix corresponding to these states. We can use this to write down a formula for the Rényi entropy of a density matrix  $\rho$  corresponding to the communication matrix  $M$  of a function (as in the previous section, specialising to the uniform distribution on Alice and Bob's inputs), which gives a lower bound on its communication capacity and thus its entanglement-assisted communication complexity.

$$S_2(\rho) = -\log \operatorname{tr} \left( \frac{1}{2^{4n}} (MM^\dagger)^2 \right) = 4n - \log \left( \sum_{i,j} \left( \sum_k M_{ik} M_{jk} \right)^2 \right) \quad (28)$$

$$= 4n - \log \left( \sum_{i,j,k,l} M_{ik} M_{jk} M_{il} M_{jl} \right) \quad (29)$$

Rényi entropic arguments have previously been used in a different way by van Dam and Hayden [7] to put lower bounds on quantum communication complexity.

### 4 The Quantum Communication Complexity of a Random Function

In this section, we will show a lower bound on the communication capacity – and thus the quantum communication complexity – of a random function (one which takes the value 0 or 1 on each possible input with equal probability). Define the state  $\rho$  as

$$\rho = \frac{1}{2^n} \sum_{k=1}^{2^n} |\psi_k\rangle\langle\psi_k|, \text{ where } |\psi_k\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (-1)^{a_{i+1}^k} |i\rangle \quad (30)$$

where  $a^k$  is a randomly generated  $2^n$ -bit string, and  $a_i^k$  represents the  $i$ 'th bit of  $a^k$ . We will show that the Rényi entropy  $S_2(\rho)$  is high for almost all  $\rho$ .

**Theorem 3.**  $\Pr[S_2(\rho) < (1 - \delta)n] \leq e^{-(2^{\delta n} - 1)^2/2}$ .

*Proof.* We have

$$S_2(\rho) = 4n - \log \left( \sum_{i,j} \left( \sum_k M_{ik} M_{jk} \right)^2 \right) \tag{31}$$

$$= 4n - \log \left( \sum_i \left( \sum_k (M_{ik})^2 \right)^2 + \sum_{i \neq j} \left( \sum_k M_{ik} M_{jk} \right)^2 \right) \tag{32}$$

$$= 4n - \log (N^3 + T) \tag{33}$$

where we define  $N = 2^n$  and  $T = \sum_{i \neq j} (\sum_k M_{ik} M_{jk})^2$ . It is then clear that

$$\Pr [S_2(\rho) < (1 - \delta)n] = \Pr [T > N^3(N^\delta - 1)] \tag{34}$$

Each term in the inner sum in  $T$  (the sum over  $k$ ) is independent and picked uniformly at random from  $\{-1, 1\}$ . We will now produce a tail bound for  $T$  using ‘‘Bernstein’s trick’’ (see Appendix A of [3]): from Markov’s inequality we have

$$\Pr [T > a] < \mathbb{E}(e^{\lambda T})/e^{\lambda a} < \mathbb{E}(e^{\lambda X_{11}})^{N^2}/e^{\lambda a} \tag{35}$$

where we define  $X_{ij} = (\sum_k M_{ik} M_{jk})^2$ : each  $X_{ij}$  is independent and identically distributed, so  $T$  is the sum of  $N(N - 1) < N^2$  copies of  $X_{11}$ . It remains to calculate  $\mathbb{E}(e^{\lambda X_{11}})$ . This can be written out explicitly as follows.

$$\mathbb{E}(e^{\lambda X_{11}}) = \frac{1}{2^N} \sum_{k=0}^N \binom{N}{k} e^{\lambda(N-2k)^2} \tag{36}$$

It is then straightforward to see (using an inequality from [3]) that the following series of inequalities holds.

$$\mathbb{E}(e^{\lambda X_{11}}) \leq \frac{1}{2^N} \sum_{k=0}^N \binom{N}{k} \left( e^{\lambda(N-2k)^2} + e^{-\lambda(N-2k)^2} \right) \tag{37}$$

$$\leq \frac{1}{2^{N-1}} \sum_{k=0}^N \binom{N}{k} e^{\lambda^2(N-2k)^4/2} \leq \frac{1}{2^{N-1}} \sum_{k=0}^N \binom{N}{k} e^{\lambda^2 N^4/2} = 2e^{\lambda^2 N^4/2} \tag{38}$$

Inserting this in (35), and minimising over  $\lambda$ , gives

$$\Pr [T > a] < 2e^{-a^2/2N^6} \tag{39}$$

and substituting  $a = N^3(N^\delta - 1)$  gives the required result.

In particular, putting  $\delta = 1/2$  gives that  $\Pr [S_2(\rho) < n/2] \leq 2e^{-(\sqrt{N}-1)^2/2}$ , which is doubly exponentially small in  $n$ . As  $\rho$  corresponds to the communication matrix of a random function, Theorem 2 immediately gives the result that the entanglement-assisted quantum communication complexity of almost all functions is  $\Omega(n)$ .

## 5 Discussion and Open Problems

We have shown that the implementation of any distributed computation between Alice and Bob entails the ability to communicate from one user to the other. This communication capacity of a Boolean function of two arguments is naturally a lower bound on the communication complexity to compute that function, and we have proved corresponding lower bounds, even in the presence of arbitrary entanglement.

These bounds show that random functions of two  $n$ -bit strings mostly have communication complexity close to  $n$ . However, in general it has to be noted that our bounds are not always tight: an example is provided by the set-disjointness problem, where Alice and Bob want to determine if their strings  $x$  and  $y$  have a position where they are both 1. It is known that the quantum communication complexity of this function is  $\Theta(\sqrt{n})$  [22,11]. On the other hand, an implicit upper bound on the entropy in our main theorem was already given for this case in [2], and it is only  $O(\log n)$ . Thus, not quite surprisingly, the ability of a function to let Alice communicate to Bob is not the same as the communication cost of implementing this computation.

Looking again at our main theorem, we are left with one interesting question: is the logarithmic factor that we lose in the bounded error model really necessary? It appears to be a technicality, since we need to boost the success probability to apply Fannes' inequality, but we were unable to determine if it is just that or if there are cases in which the lower bound is tight.

**Acknowledgements.** AM would like to thank Richard Jozsa for careful reading and comments on this manuscript, and Tony Short and Aram Harrow for helpful discussions. We thank Ronald de Wolf for pointing out references [17] and [19], and a referee for helpful comments on presentation. AW acknowledges support via the EC project QAP, as well as from the U.K. EPSRC. He also gratefully notes the hospitality of the Perimeter Institute for Theoretical Physics in Waterloo, Ontario, where part of this work was done.

## References

1. Aaronson, S., Ambainis, A.: Quantum search of spatial regions. Theory of Computing, vol 1, pp. 47-79 (2005), [quant-ph/0303041](#)
2. Ambainis, A., Schulman, L.J., Ta-Shma, A., Vazirani, U., Wigderson, A.: The quantum communication complexity of sampling. SIAM J. Comput. 32, 1570–1585 (2003)
3. Alon, N., Spencer, J.: The probabilistic method. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, New York (2000)
4. Buhrman, H., de Wolf, R.: Communication complexity lower bounds by polynomials. In: Proc. CCC'01, pp. 120-130 (2001), [cs.CC/9910010](#)
5. Cleve, R., van Dam, W., Nielsen, M., Tapp, A.: Quantum entanglement and the communication complexity of the inner product function. In: Selected papers from the First NASA International Conference on Quantum Computing and Quantum Communications, pp. 61–74 (February 17-20 1998), [quant-ph/9708019](#)

6. Fannes, M.: A continuity property of the entropy density for spin lattice systems. In: Commun. Math. Phys. vol. 31, pp. 291–294 (1973)
7. van Dam, W., Hayden, P.: Renyi-entropic bounds on quantum communication (2002), [quant-ph/0204093](#)
8. Gavinsky, D., Kempe, J., de Wolf, R.: Strengths and weaknesses of quantum fingerprinting. In: Proc. CCC'06, pp. 288–298 (2006), [quant-ph/0603173](#)
9. Hausladen, P., Jozsa, R., Schumacher, B., Westmoreland, M., Wootters, W.: Classical information capacity of a quantum channel. Phys. Rev. A 54(3), 1869–1876 (1996)
10. Holevo, A.S.: Bounds for the quantity of information transmittable by a quantum communications channel. Problemy Peredachi Informatsii, vol. 9(3), pp. 3–11, 1973. English translation Problems of Information Transmission, vol. 9, pp. 177–183 (1973)
11. Horn, R.A., Johnson, C.: Matrix analysis. Cambridge University Press, Cambridge (1996)
12. Jozsa, R., Schlienz, J.: Distinguishability of states and von Neumann entropy. Phys. Rev. A 62 012301 (2000), [quant-ph/9911009](#)
13. Klauck, H.: Lower bounds for quantum communication complexity. Proc. FOCS'01, pp. 288–297 (2001), [quant-ph/0106160](#)
14. Kremer, I.: Quantum communication. Master's thesis, Hebrew University (1995)
15. Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press, Cambridge (1997)
16. Linial, N., Shraibman, A.: Learning complexity vs. communication complexity. Manuscript (2006), <http://www.cs.huji.ac.il/~nati/PAPERS/lcc.pdf>
17. Linial, N., Shraibman, A.: Lower bounds in communication complexity based on factorization norms. In: Proc. STOC'07, (to appear 2007) [http://www.cs.huji.ac.il/~nati/PAPERS/quant\\_cc.pdf](http://www.cs.huji.ac.il/~nati/PAPERS/quant_cc.pdf)
18. Nayak, A., Salzman, J.: On communication over an entanglement-assisted quantum channel. In: Proc. STOC'02, pp. 698–704 (2002), [quant-ph/0206122](#)
19. Nielsen, M.A.: Quantum information theory. PhD thesis, University of New Mexico, Albuquerque (1998), [quant-ph/0011036](#)
20. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge University Press, Cambridge (2000)
21. Raz. Exponential separation of quantum and classical communication complexity. In: Proc. STOC'99, pp. 358–367 (1999)
22. Razborov, A.A.: Quantum communication complexity of symmetric predicates. Izvestiya of the Russian Academy of Science, vol. 67, pp. 159–176 (2003), [quant-ph/0204025](#)
23. Probability, A.R.: Probability theory. North-Holland, Amsterdam (1970)
24. de Wolf, R.: Quantum communication and complexity. Theoretical Computer Science 287(1), 337–353 (2002)
25. Yao, A.: Some complexity questions related to distributive computing. In: Proc. STOC'79, pp. 209–213 (1979)

# Separating Deterministic from Nondeterministic NOF Multiparty Communication Complexity (Extended Abstract)

Paul Beame<sup>1,\*</sup>, Matei David<sup>2,\*\*</sup>, Toniann Pitassi<sup>2,\*\*\*</sup>, and Philipp Woelfel<sup>2,†</sup>

<sup>1</sup> University of Washington

<sup>2</sup> University of Toronto

**Abstract.** We solve some fundamental problems in the number-on-forehead (NOF)  $k$ -party communication model. We show that there exists a function which has at most logarithmic communication complexity for randomized protocols with a one-sided error probability of  $1/3$  but which has linear communication complexity for deterministic protocols. The result is true for  $k = n^{O(1)}$  players, where  $n$  is the number of bits on each players' forehead. This separates the analogues of RP and P in the NOF communication model. We also show that there exists a function which has constant randomized complexity for public coin protocols but at least logarithmic complexity for private coin protocols. No larger gap between private and public coin protocols is possible. Our lower bounds are existential and we do not know of any explicit function which allows such separations. However, for the 3-player case we exhibit an explicit function which has  $\Omega(\log \log n)$  randomized complexity for private coins but only constant complexity for public coins.

It follows from our existential result that any function that is complete for the class of functions with polylogarithmic nondeterministic  $k$ -party communication complexity does not have polylogarithmic deterministic complexity. We show that the set intersection function, which is complete in the number-in-hand model, is not complete in the NOF model under cylindrical reductions.

## 1 Introduction

The question of how much communication is necessary in order to compute a function  $f : X_1 \times \cdots \times X_k \rightarrow O$  when its input is distributed between  $k$  computationally unbounded players was first introduced in [17] and it has since been shown to have many diverse applications in complexity theory. The case of  $k = 2$  players has been studied extensively [11]. For two or more players, we are interested in the "number-on-forehead" model (NOF), first introduced by

---

\* Supported by NSF grant CCR-0514870.

\*\* Supported by OGS.

\*\*\* Supported by NSERC.

† Supported by DFG grant Wo 1232/1-1.

Chandra, Furst and Lipton in [7]. In this model, the input is partitioned into  $k$  parts, so that player  $i$  can see all parts except for the  $i^{\text{th}}$  part (since it is ‘written on his forehead’).

The number-on-forehead communication model is a fascinating and complex model that is not well understood when  $k \geq 3$ . The complexity of the situation arises from the fact that every part of the input is seen by multiple players. As the number of players increases, the sharing becomes increasingly generous. During the execution of a protocol, the set of inputs consistent with a particular message sequence is described by a so-called cylinder intersection. Cylinder intersections appear difficult to understand combinatorially.

Lower bounds for multiparty complexity in the number-on-forehead model are connected to a major open problem in complexity theory: it has been established that superlogarithmic communication complexity lower bounds in the NOF model for any explicit function with polylogarithmically many players would imply explicit lower bounds for ACC [6, 10]. The best lower bound obtained so far establishes a lower bound of  $\Omega(n/2^k)$ , which breaks down when the number of players is greater than logarithmic [3, 8, 16, 9]. Lower bounds in this model have many other important applications as well, including: constructions of pseudorandom generators for space bounded computation, constructions of universal traversal sequences, time-space tradeoffs [3], circuit complexity bounds [10, 15, 14], and proof complexity bounds [4].

The motivation for our work is to pursue a broader understanding of the NOF complexity model. In particular, we would like to answer some of the basic questions that are still open for this model, but have well-known solutions in the 2-party model. For  $k \geq 3$ , we consider the three usual versions of communication complexity: deterministic, randomized and nondeterministic complexity. Are there functions separating these three different complexity measures? Surprisingly, the relationships between these complexity measures have not been resolved previously, even for  $k = 3$ .

Our main result is that for any  $k$  that is  $n^{O(1)}$  there is a function with  $n$  bits on each players’ forehead that is computable with a polylogarithmic complexity by a randomized  $k$ -party communication protocol with 1-sided error but which requires linear complexity for deterministic protocols. We obtain this result nonconstructively by showing that deterministic protocols for a certain class of *simple* functions have a nice normal form and then establishing a lower bound for such function via a counting argument over protocols in normal form. We thus separate the randomized 1-sided error and deterministic  $k$ -party NOF communication complexity classes  $\text{RP}_k^{cc}$  and  $\text{P}_k^{cc}$ . As a corollary of our lower bounds, we also establish an optimal separation between the public and private coin randomized NOF models.

These bounds are nonconstructive but, for  $k$  at most logarithmic in the input size, we can also give *explicit* families of simple functions with  $\Omega(\log n)$  deterministic  $k$ -party complexity in the NOF model. (We believe that they have superpolylogarithmic deterministic complexity.) The best previous lower bound for any explicitly defined simple function is the  $\Omega(\log \log n)$  lower bound from [5]



for the Exact-T function (originally investigated in [7]) in the special case of  $k = 3$  players. As a corollary of our bound we obtain that our function families have  $\Omega(\log \log n)$  complexity for randomized private coin protocols (with constant error probability) but only  $O(1)$  complexity for public coin protocols.

The problem of separating deterministic from nondeterministic NOF complexity is particularly interesting because of its connection to proof complexity. In recent work [4], it has been shown that for  $k = 3$ ,  $(\log n)^{\omega(1)}$  lower bounds on the randomized NOF complexity of set intersection, which has nondeterministic NOF complexity  $O(\log n)$ , implies lower bounds for polynomial threshold proof systems, such as the Lovász-Schrijver proof systems, as well as the Chvátal cutting planes proof system. Moreover, it seems possible that these results can be modified to show that randomized lower bounds for *any* function with small NOF nondeterministic communication complexity would give lower bounds for related cutting planes proof systems.

This brings us to our second question: is there a ‘complete’ problem for the class of problems with efficient NOF nondeterministic algorithms under a suitable notion of reduction? Given our separation result, such a function would automatically be hard for deterministic protocols. Following [1], it is not hard to see that set intersection is complete under communication-free reductions for the number-in-hand (NIH) model and in [4] it had been assumed that the same holds for the number-on-forehead (NOF) model. (The number-in-hand model is an alternative generalization of the 2-player model in which each player gets his part of the input in his hand, and thus each player sees only his own part.) However, we prove that under communication-free reductions, set intersection is not complete in the NOF model.

## 2 Definitions and Preliminaries

In the NOF multiparty communication complexity game [7] there are  $k$  parties (or players), numbered 1 to  $k$ , that are trying to collaborate to compute a function  $f : X_1 \times \dots \times X_k \rightarrow \{0, 1\}$  where each  $X_i = \{0, 1\}^n$ . The  $kn$  input bits are partitioned into  $k$  sets, each of size  $n$ . For  $(x_1, \dots, x_k) \in \{0, 1\}^{kn}$ , and for each  $i$ , player  $i$  knows the values of all of the inputs except for  $x_i$  (which conceptually is thought of as being placed on player  $i$ ’s forehead).

The players exchange bits according to an agreed-upon protocol, by writing them on a public blackboard. A *protocol* specifies, for every possible blackboard contents, whether or not the communication is over, the output if over and the next player to speak if not. A protocol also specifies what each player writes as a function of the blackboard contents and of the inputs seen by that player. The *cost* of a protocol is the maximum number of bits written on the blackboard.

In a *deterministic* protocol, the blackboard is initially empty. A *public-coin randomized* protocol of cost  $c$  is simply a probability distribution over deterministic protocols of cost  $c$ , which can be viewed as a protocol in which the players have access to a shared random string. A *private-coin randomized* protocol is a protocol in which each player has access to a private random string.

A *nondeterministic* protocol is a randomized private coin protocol with 1-sided error (only false negatives) and an error probability less than 1.

The *deterministic* communication complexity of  $f$ , written  $D_k(f)$ , is the minimum cost of a deterministic protocol for  $f$  that always outputs the correct answer. For  $0 \leq \epsilon < 1/2$ , let  $R_{k,\epsilon}^{\text{pub}}(f)$  denote the minimum cost of a public-coin randomized protocol for  $f$  which, for every input, makes an error with probability at most  $\epsilon$  (over the choice of the deterministic protocols). We write  $R_k^{\text{pub}}(f)$  for  $R_{k,1/3}^{\text{pub}}(f)$ . Let  $R_{k,\epsilon}(f)$  denote the minimum cost of a private-coin randomized protocol for  $f$  which, for every input, makes an error with probability at most  $\epsilon$  (over the choice of the private random strings). We write  $R_k(f)$  for  $R_{k,1/3}(f)$ . For both public-coin and private-coin complexities we add a superscript 1 if we require that the protocol makes error only on 1-inputs (i.e., false-negatives), and superscript 0 if we require that the protocol makes error only on 0-inputs (i.e., false-positives). For example,  $R_{k,\epsilon}^{0,\text{pub}}(f)$  is the minimum cost of a  $k$ -player public-coin protocol for  $f$  which is always correct on 1-inputs and makes error at most  $\epsilon$  on 0-inputs.

Since the general model laid out above is very powerful, we are also interested in communication restrictions. A player is *oblivious* in a certain protocol if the message he writes on the board is a function of the inputs he sees, but not a function of the messages sent by other players. Since we are interested in the best protocol, we may safely assume that all oblivious players write first, and then non-oblivious players continue to communicate using the information written by the former. A protocol in which all players are oblivious is called *simultaneous*. The simultaneous multiparty model was studied in [2], who proved new lower bounds, as well as surprising upper bounds in this model.

Since any function  $f_n$  on  $kn$  bits can be computed using only  $n$  bits of communication, following [1], for sequences of functions  $f = (f_n)_{n \in \mathbb{N}}$ , algorithms are considered “efficient” or “polynomial” if only polylogarithmically many bits are exchanged. Accordingly, let  $\mathbf{P}_k^{\text{cc}}$  denote the class of function families  $f$  for which  $D_k(f_n)$  is  $(\log n)^{O(1)}$ , let  $\mathbf{NP}_k^{\text{cc}}$  denote the class of function families  $f$  with nondeterministic complexity  $(\log n)^{O(1)}$ , and let  $\mathbf{RP}_k^{\text{cc}}$  denote the class of function families  $f$  for which  $R_k^1(f_n)$  is  $(\log n)^{O(1)}$ .

Multiparty communication complexity lower bounds are proven by analyzing properties of functions on *cylinder intersections*.

**Definition 1.** An  $i$ -cylinder  $C_i$  in  $X_1 \times \dots \times X_k$  is a set such that for all  $x_1 \in X_1, \dots, x_k \in X_k, x'_i \in X_i$  we have  $(x_1, \dots, x_i, \dots, x_k) \in C_i$  if and only if  $(x_1, \dots, x'_i, \dots, x_k) \in C_i$ . A cylinder intersection is a set of the form  $\bigcap_{i=1}^k C_i$  where each  $C_i$  is an  $i$ -cylinder in  $X_1 \times \dots \times X_k$ .

### 3 Separating $\mathbf{P}_k^{\text{cc}}$ from $\mathbf{RP}_k^{\text{cc}}$

#### 3.1 Oblivious Players, Simple Functions, and a Normal Form

We will be interested in a special type of Boolean functions for which we can show, that without loss of generality, all but one of the players is oblivious. For

sets  $X_1, \dots, X_k$  a function  $f : X_1 \times \dots \times X_k \rightarrow \{0, 1\}$  is *simple for player  $i$*  if for all  $(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k) \in X_1 \times \dots \times X_{i-1} \times X_{i+1} \times \dots \times X_k$  there exists at most one  $x_i^* \in X_i$  such that  $f(x_1, \dots, x_{i-1}, x_i^*, x_{i+1}, \dots, x_k) = 1$ .

If  $f$  is simple for player  $i$  then it is reducible with no communication to 2-player  $n$ -bit equality EQ. Player  $i$  can compute the unique value for the input on its forehead for which the output could be 1 (if it exists), and any other player sees that input. All the players have to do is to decide whether these strings are equal. We know that  $R_{2,1/n}^0(\text{EQ})$  is  $O(\log n)$  and  $R_2^{0,\text{pub}}(\text{EQ})$  is  $O(1)$ . Therefore we get the following.

**Lemma 2.** *For all  $k$  and all simple functions  $f$  on  $kn$  bits,  $R_{k,1/n}^0(f)$  is  $O(\log n)$  and  $R_k^{0,\text{pub}}(f)$  is  $O(1)$ .*

The following theorem shows that if a function is simple for one player then this player can act obliviously with only a small increase in the deterministic communication complexity.

**Theorem 3.** *Let  $f : X_1 \times \dots \times X_k \rightarrow \{0, 1\}$  be a function that is simple for player  $i$  and has  $D_k(f) = d$ . Then there is a protocol  $P'$  for  $f$  in which player  $i$  first sends  $d$  bits and then all players  $j \in \{1, \dots, k\} - \{i\}$  simultaneously send exactly one bit  $b_j$  such that  $f(x_1, \dots, x_k) = 1$  if and only if all bits  $b_j = 1$ .*

*Proof (Sketch).* Let  $f$  be simple for player 1. Let  $P$  be a protocol for  $f$  with complexity  $d$ . We describe protocol  $P'$  on input  $(x_1, \dots, x_k)$ . Assume that player 1 sees the partial input  $(x_2, \dots, x_k)$  on the other players' foreheads. Let  $x_1^*$  be the input in  $X_1$  such that  $f(x_1^*, x_2, \dots, x_k) = 1$ , if one exists, an arbitrary input in  $X_1$ , otherwise. Player 1 “simulates” protocol  $P$  for the input  $(x_1^*, x_2, \dots, x_k)$ ; i.e., she writes on the blackboard exactly the string  $I^*$  that would have been written by players  $1, \dots, k$  if protocol  $P$  were executed for that input. Then each player  $r$ ,  $2 \leq r \leq k$ , verifies that  $I^*$  is consistent with what player  $r$  would have sent in protocol  $P$  if it had seen  $(x_1, \dots, x_{r-1}, x_{r+1}, \dots, x_k)$  on the other players' foreheads. If player  $r$  does not find an error and the output of  $P$  is 1 for blackboard contents  $I^*$ , then he accepts by sending bit  $b_r = 1$ . Otherwise he sends  $b_r = 0$ . □

### 3.2 Representing Simple Functions by Colorings and Cylinder Intersections

Most lower bound proofs for  $D_k(f)$  use the fact shown in [3] that any  $k$ -party protocol with complexity  $d$  for a function  $f$  yields a partitioning of the input into  $O(2^d)$  disjoint cylinder intersections on which  $f$  is constant. For  $k \geq 3$  players, the known techniques for proving lower bounds on the number of cylinder intersections needed for such a partitioning are discrepancy-based and inherently yield lower bounds even for nondeterministic and randomized protocols. Therefore, these techniques are not suitable for proving good lower bounds for functions with low nondeterministic communication complexity.

For simple functions we obtain different, although related, structures. These structures seem to be better suited for lower bound proofs for functions in  $\mathbf{RP}_k^{cc}$ , as they will allow us to separate this class from  $\mathbf{P}_k^{cc}$  and to prove  $\Omega(\log n)$  lower bounds for explicit functions.

Throughout this section,  $f : X_1 \times \cdots \times X_k \rightarrow \{0, 1\}$  is simple for player 1. For any natural number  $D$  and a set  $S$ , a  $D$ -coloring of  $S$  is a mapping  $c : S \rightarrow [D]$ . Since  $f$  is simple for player 1 (Alice), there exists a function  $g : X_2 \times \cdots \times X_k \rightarrow X_1 \cup \{\perp\}$ , where  $g(x_2, \dots, x_k) = \perp$  if  $f(x_1, \dots, x_k) = 0$  for all  $x_1 \in X_1$ , and otherwise  $g(x_2, \dots, x_k) = x_1^*$ , where  $x_1^*$  is the unique element in  $X_1$  with  $f(x_1^*, x_2, \dots, x_k) = 1$ . In fact, any such mapping  $g$  uniquely defines the simple function  $f$ .

Assume that  $f$  can be computed by a  $d$ -bit protocol  $P$ . The special protocol  $P'$  for  $f$ , derived in Theorem 3, can be characterized by a coloring of  $X_2 \times \cdots \times X_k$  and cylinder intersections in  $X_2 \times \cdots \times X_k$ : Let  $c$  be the  $2^d$ -coloring of  $X_2 \times \cdots \times X_k$ , where  $c(x_2, \dots, x_k)$  is the message Alice sends if she sees  $(x_2, \dots, x_k)$ . Consider a fixed message  $m$  from Alice and a fixed value  $a \in X_1$  on Alice's forehead. The subset of points in  $X_2 \times \cdots \times X_k$  for which all other players accept if they see  $a$  on Alice's forehead and receive message  $m$  is a cylinder intersection  $I_{m,a}$ . Each such cylinder intersection  $I_{m,a}$  may also contain points that are not colored  $m$ . However, it is not possible that a point  $p = (x_2, \dots, x_k) \in I_{m,a}$  has color  $m$  but  $g(p) \neq a$  because then Alice would send message  $m$  if she saw  $p$  and the other players would all accept if they saw  $a$  on Alice's forehead. Hence,  $(a, x_2, \dots, x_k)$  would be accepted by  $P'$ , a contradiction. We obtain the following.

**Lemma 4.** *Every function  $f$  that is simple for player 1 and has  $k$ -player communication complexity  $d$  can be uniquely represented by cylinder intersections  $I_{m,a} \in X_2 \times \cdots \times X_k$ ,  $m \in [2^d]$ ,  $a \in X_1$ , and a  $2^d$ -coloring  $c$  of  $X_2 \times \cdots \times X_k$ , such that  $\forall a \in X_1, y \in X_2 \times \cdots \times X_k$ :  $f(a, y) = 1 \Leftrightarrow y \in I_{c(y),a}$ .*

*In particular,  $I_{m,a}$  contains all points  $y \in X_2 \times \cdots \times X_k$  with color  $c(y) = m$  and  $f(a, y) = 1$ , but no point  $y'$  with color  $c(y') = m$  and  $f(a, y') = 0$ .*

*Proof.* We have already seen how to obtain  $c$  and the cylinder intersections  $I_{m,a}$  from the function  $f$ . This representation is unique because for any input  $(a, p)$  with  $a \in X_1$  and  $p \in X_2 \times \cdots \times X_k$  we can retrieve the function value  $f(a, p)$  by checking whether  $p \in I_{c(p),a}$ . □

### 3.3 The Lower Bound

In the following we consider a family of functions which have logarithmic communication complexity for randomized protocols with one-sided error and error probability bounded by  $1/3$ . Using Lemma 4 we give an upper bound on the number of different deterministic protocols for the functions in that class in order to show that at least one such function requires at least linear deterministic communication complexity.

For positive integers  $n, m$  and  $t$ , let  $G_{t,n,m}$  be the set of all mappings  $g : \{0, 1\}^{n-t} \rightarrow \{0, 1\}^m$ . For any function  $g \in G_{k-1,n,m}$ , define  $f_g : \{0, 1\}^n \times \{0, 1\}^{n-(k-1)}$  by  $f_g(x_1, \dots, x_k) = 1$  if and only if  $g(x_2, \dots, x_k) = x_1$ . By the proof

of Lemma 2, randomized protocols for functions  $f_g, g \in G_{k,n,m}$ , have complexity at most  $O(\log m)$ . Hence, it follows that  $f_g \in \text{co-RP}_k^{\text{cc}}$  for all  $g \in G_{k,n,n/2}$ .

**Theorem 5.** *There is a  $g \in G_{k-1,n,n/2}$  such that  $D_k(f_g)$  is  $\Omega(n - \log k)$ .*

**Corollary 6.**  $\text{P}_k^{\text{cc}} \neq \text{RP}_k^{\text{cc}}$  for any  $k$  that is  $n^{O(1)}$ .

*Proof (of Theorem 5).* Any function  $g \in G_{k-1,n,m}$  has a domain of size  $2^{(k-1)n}$  and a range of size  $2^m$ . Therefore, it is not possible to encode every such function  $g$  with less than  $m \cdot 2^{(k-1)n}$  bits. Note that if two functions  $g, g'$  are different, then  $f_g$  and  $f_{g'}$  are different, too.

Clearly, any function  $f_g, g \in G_{k-1,n,m}$ , is simple for Alice. Assume that any such function  $f_g$  has  $D_k(f_g) \leq d$ . Then by Lemma 4 every such function  $f_g$  can be uniquely represented by a  $2^d$ -coloring of  $(\{0, 1\}^n)^{k-1}$  and  $2^m \cdot 2^d$  cylinder intersections in  $(\{0, 1\}^n)^{k-1}$ . The  $2^d$ -coloring of  $(\{0, 1\}^n)^{k-1}$  can be encoded with  $d \cdot 2^{(k-1)n}$  bits. The number of  $i$ -cylinders in  $X_1 \times \dots \times X_t$  is  $2^{\prod_{j \neq i} |X_j|}$ . Hence,  $(k-1) \cdot 2^{(k-2)n}$  bits suffice for a unique encoding of any cylinder intersection in  $(\{0, 1\}^n)^{k-1}$ . Thus, the total number of bits in which any function  $f_g, g \in G_{k-1,n,m}$ , can be encoded is bounded above by

$$d \cdot 2^{(k-1)n} + 2^{d+m} \cdot (k-1) \cdot 2^{(k-2)n} = d \cdot 2^{(k-1)n} + (k-1) \cdot 2^{d+m+(k-2)n}$$

As we have seen above, the number of bits needed to describe a function  $f_g$  for  $g \in G_{k-1,n,m}$  is at least  $m \cdot 2^{(k-1)n}$ . Therefore, if for all  $f_g$  a protocol with complexity  $c$  exists then

$$d \cdot 2^{(k-1)n} + (k-1) \cdot 2^{d+m+(k-2)n} \geq m \cdot 2^{(k-1)n}.$$

This is equivalent to  $2^d \geq 2^{n-m} \cdot (m-d)/(k-1)$ . Hence,  $d \geq \min\{m-1, n-m-\log(k-1)\}$ , which for  $m = \lfloor (n - \log k)/2 \rfloor$  is at least  $(n - \log k)/2 - O(1)$ .  $\square$

### 3.4 Separating Public from Private Coins

We now consider the difference between public-coin and private-coin randomized protocols. Trivially, any private-coin protocol can be simulated by tossing the coins in public, so for all  $f$  and  $k$ ,  $R_k^{\text{pub}}(f) \leq R_k(f)$ . In the other direction, Newman [13, 11] provides a simulation of a public-coin protocol by a private-coin protocol. (Although it is stated for the special case of 2 players, the proof works for any number of players.)

**Proposition 7 ([13]).** *There is a  $c > 0$  such that for every  $k \geq 2$  and function  $f : \{0, 1\}^{kn} \rightarrow \{0, 1\}$ ,  $R_k(f) \leq R_k^{\text{pub}}(f) + c \lceil \log_2 n \rceil$ .*

We see that the maximum gap between the public-coin and private-coin randomized complexities of  $f$  is  $\Theta(\log n)$ , and it is achieved when  $R_k^{\text{pub}}(f)$  is  $O(1)$  and  $R_k(f)$  is  $\Theta(\log n)$ . The natural question arises, is there a function that achieves this gap? Our results allow us to answer this question affirmatively.

In order to obtain lower bounds, we need the following extension of Lemma 3.8 in [11] to  $k$  players. We omit the proof due to space constraints.

**Lemma 8.** *If  $k^{1/\delta} < D_k(f)$  for some  $\delta < 1$ , then  $R_k(f)$  is  $\Omega(\log D_k(f))$ .*

**Corollary 9.** *Let  $\delta < 1$ . For all  $k$  such that  $k < n^\delta$ , there exists a  $kn$ -bit function  $f$  such that  $R_k^{\text{pub}}(f)$  is  $O(1)$  and  $R_k(f)$  is  $\Theta(\log n)$ .*

*Proof.* By Theorem 5 there is a function  $f$  that is simple for player 1 such that  $D_k(f)$  is  $\Omega(n)$ . By Lemma 8,  $R_k(f)$  is  $\Omega(\log n)$ . By Lemma 2,  $R_k(f)$  is  $O(\log n)$  and  $R_k^{\text{pub}}(f)$  is  $O(1)$ . □

## 4 Lower Bounds for Explicit Simple Functions

The separations in Section 3.3 are nonconstructive. We conjecture that there also exists an *explicit* simple function that gives a linear or near linear separation between  $D_k(f_n)$  and  $R_k^0(f_n)$ . In the following we prove  $\Omega(\log n)$  bounds for the deterministic complexity of some explicit simple functions. This yields a separation between the deterministic and public coin randomized complexity for explicit simple functions, though this is much weaker than our conjecture.

We give two constructions of explicit functions, one for  $k = 3$  and, one that holds for all  $k \geq 3$ . Write  $\mathbb{F}_q$  for the field of  $q$  elements. Let  $X = \mathbb{F}_{2^n}$ ,  $Y = \mathbb{F}_{2^m}$  for some positive integers  $m$  and  $n$ . For  $(a, b) \in X \times Y$  let the hash function  $h_{a,b} : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^m}$  be defined as  $h_{a,b}(x) = \phi(a \cdot x) + b$ , where  $\phi$  is a homomorphism from  $\mathbb{F}_{2^n}$  to  $\mathbb{F}_{2^m}$ . Let  $H$  be the family of hash functions  $h_{a,b}$  for  $a \in X$ ,  $b \in Y$ . The explicit function for  $k = 3$  is given by  $f : Y \times X \times H \rightarrow \{0, 1\}$ , where  $f(y, x, h) = 1$  iff  $h(x) = y$  which is clearly simple for player 1.

For the other construction let  $k \geq 3$ , let  $X_1 = \mathbb{F}_{2^m}$  and  $X_2 = \dots = X_k = \mathbb{F}_{2^n}^n$  for positive integers  $n$  and  $m$  with  $m \geq \log_2 n$ . Let  $\beta_1, \dots, \beta_n$  be distinct elements of  $\mathbb{F}_{2^m}$  and define  $v_i = (\beta_1^{i-1}, \dots, \beta_n^{i-1})$  for  $1 \leq i \leq n$ . The explicit function is  $f_g$  where  $g(x_2, \dots, x_k) = \sum_{i=1}^n \prod_{j=2}^k \langle v_i, x_j \rangle$  and operations are over  $\mathbb{F}_{2^m}$ .

**Theorem 10.** *There is a  $\delta < 1$  such that*

- (a) *for  $m = n^\delta$  and  $f : Y \times X \times H \rightarrow \{0, 1\}$  defined as above,  $D_3(f)$  is  $\Omega(\log n)$ ,*
- (b) *for any  $k \geq 3$ ,  $n \geq 4^{k+1}$ ,  $m = n^\delta$ , and  $f_g$  defined as above,  $D_k(f_g)$  is  $\Omega(\log n)$ .*

*Proof.* We give the proof for part (a): Let  $d = D_3(f)$ . Fix some  $\epsilon$  with  $0 < \epsilon < \delta$  and let  $m = n^\delta$ . Assume for contradiction that  $d \leq \epsilon \cdot \log_2 n$ .

Consider the  $2^{n+m} \times 2^n$  matrix  $M$  where rows correspond to hash functions  $h \in H$ , columns correspond to inputs  $x \in X$  and the entry  $M_{h,x}$  is the hash function value  $h(x)$ . Cylinder intersections in  $H \times X$  are rectangles. By Lemma 4, there is a  $2^d$ -coloring  $c$  of  $M$  and there are  $2^{d+m}$  rectangles  $R_{\ell,y}$ ,  $\ell \in [2^d]$ ,  $y \in Y$ , such that  $\forall (y, x, h) \in Y \times X \times H : (h, x) \in R_{c(h,x),y} \Leftrightarrow h(x) = y$ . Call an entry  $M_{h,x}$  an  $(\ell, y)$  entry iff  $c(h, x) = \ell$  and  $h(x) = y$ . Correctness of the protocol implies that for  $y' \neq y$ ,  $R_{\ell,y}$  does not contain any  $(\ell, y')$  entries.

Consider the coloring  $c$  of the matrix entries in  $M$ . The proof proceeds inductively decreasing the number of colors that are available and shrinking the matrix. During each such step, we introduce a number of “holes” in the matrix (entries that are colored in the original matrix with one of the removed colors). We show that eventually there are no colors left to use but the matrix still does

not consist only of holes. This will contradict the existence of the initial coloring, hence of the  $d$ -bit protocol.

We prove by induction on  $i \geq 0$  that, as long as  $i \leq 2^d = n^\epsilon$ , the following hold for large enough  $n$ :

- there exists a rectangle  $R_i$  such that  $|R_i| \geq 2^{2n+m-i(d+2)}$ ,
- $R_i$  contains at most  $i \cdot 2^{2n}$  holes,
- non-hole entries in  $R_i$  can be colored with  $2^d - i$  colors.

Assuming that we have established this inductive statement, letting  $i = 2^d$  we see that there are no colors left for coloring the rectangle  $R_{2^d}$ . Moreover, for large enough  $n$ , this rectangle has size at least  $2^{2n+m-n^\epsilon(2+\epsilon \cdot \log_2 n)}$ . Since  $m = n^\delta$  and  $\delta > \epsilon$ ,  $|R_{2^d}| > 2^{2n+d}$ . The number of holes in this rectangle is bounded above by  $2^d \cdot 2^{2n}$ , so the rectangle is not empty, which is a contradiction.

We now prove the inductive statement. For  $i = 0$ , let  $R_0 = M$ . The existence of a  $d$ -bit protocol yields a coloring of  $R_0$  (with no holes) using  $2^d$  colors.

Now assume the inductive statement is true for some  $0 \leq i < 2^d$ . The number of non-hole entries in  $R_i$  is at least  $2^{2n+m} \cdot (2^{-i(d+2)} - i \cdot 2^{-m})$ . Since  $i < 2^d$  and  $m - d > n^\epsilon(d+2) > i(d+2)$  (for large enough  $n$ ), the number of non-hole entries in  $R_i$  is larger than  $2^{2n+m-i(d+2)-1}$ . Let  $(\ell, y)$  be the most popular color-value pair from the non-hole entries in  $R_i$  and let  $R_{i+1} = R_i \cap R_{\ell, y}$ . The number of color-value pairs is at most  $2^{d+m}$ , so the number of occurrences of the most popular pair  $(\ell, y)$  in  $R_i$  is at least  $2^{2n+m-i(d+2)-1-(m+d)} = 2^{2n-(i+1)(d+2)+1}$ . By construction,  $|R_{i+1}|$  is at least the number of such  $(\ell, y)$  entries. Since  $R_{i+1}$  is a rectangle, by the Hash Mixing Lemma [I2], for any  $y \in Y$ ,

$$\Pr[h(x) = y] \leq \frac{1}{|Y|} + \sqrt{\frac{|H|}{|R_{i+1}| \cdot |Y|}} \leq 2^{-m} + 2^{((i+1)(d+2)-n-1)/2} \leq 2^{-m+1}$$

since  $|H|/|R_{i+1}||Y| \leq 2^{n-2n+(i+1)(d+2)-1}$ ,  $(i+1)(d+2) \leq n^\epsilon \log n < n^\delta = m$  and  $n \geq 3m$  for sufficiently large  $n$ . Hence, the number of  $y$ -valued entries in  $R_{i+1}$  is at most  $|R_{i+1}| \cdot 2^{-m+1}$ . By the lower bound from above for the number of  $(\ell, y)$ -pairs in  $R_{i+1}$ , we have  $2^{2n-(i+1)(d+2)+1} \leq |R_{i+1}| \cdot 2^{-m+1}$  and thus we obtain the stronger bound  $|R_{i+1}| \geq 2^{2n+m-(i+1)(d+2)}$  as required.

Since  $R_{i+1} \subseteq R_{\ell, y}$ , by Lemma 4 all  $\ell$ -colored entries in  $R_{i+1}$  are  $(\ell, y)$  entries. Define the holes in  $R_{i+1}$  to be its  $(\ell, y)$  entries along with all holes in  $R_i$ . Thus, the number of colors available for non-hole entries has been reduced by at least 1. The number of extra holes we introduce is at most the number of entries in  $M$  with value  $y$ . Hence, at most  $2^{2n}$  new holes can be introduced in a round. This completes the inductive step, and therefore the proof of (a).

The proof for part (b) is similar but requires the following property of  $g$  which is a natural analogue of the Hash Mixing Lemma [I2] over cylinder intersections. Its proof is in the full paper.

**Lemma 11.** *Let  $Z = \mathbb{F}_{2^m}^n$ . For  $k \geq 3$  and any cylinder intersection  $I \subseteq Z^{k-1}$  and any  $y \in \mathbb{F}_{2^m}$ , choosing  $(x_2, \dots, x_k)$  from the uniform distribution on  $Z^{k-1}$ ,  $|\Pr[g(x_2, \dots, x_k) = y \text{ and } (x_2, \dots, x_k) \in I] - 2^{-m}|I|/|Z|^{k-1}| \leq 2^{-(m-2)n/4^{k-1}}$ .  $\square$*

By Lemma 2, both  $f$  and  $f_g$  defined above have  $O(1)$  public-coin randomized complexity but by Theorem 10 and Lemma 8, we obtain that  $R_3(f)$  and  $R_k(f_g)$  are both  $\Omega(\log \log n)$ . In fact, we conjecture that the  $D_3(f)$  and  $D_k(f_g)$  are both  $\omega(\log n)$  or even  $n^{\Omega(1)}$ . Proving the latter would yield explicit examples of function in  $\text{RP}_3^{\text{cc}}$  but not in  $\text{P}_3^{\text{cc}}$ .

### 5 On Complete Problems for $\text{NP}_k^{\text{cc}}$

An alternative approach to separating  $\text{P}_k^{\text{cc}}$  from  $\text{RP}_k^{\text{cc}}$  with an explicit function is to find a function that is complete in some sense. If we can prove for some explicit function that it is “at least as hard” as any function in  $\text{RP}_k^{\text{cc}}$ , then by our separation result we can conclude that it is not in  $\text{P}_k^{\text{cc}}$ . Proving a lower bound for a function complete for  $\text{NP}_k^{\text{cc}}$  has the added benefit of potentially separating  $\text{NP}_k^{\text{cc}}$  from  $\text{RP}_k^{\text{cc}}$  as well. (Recall that simple functions are in  $\text{RP}_k^{\text{cc}}$  so they cannot separate  $\text{NP}_k^{\text{cc}}$  from  $\text{RP}_k^{\text{cc}}$ .) The set intersection function is complete for the class analogous to  $\text{NP}_k^{\text{cc}}$  in the number-in-hand (NIH) model, and thus also for  $\text{NP}_2^{\text{cc}}$ . In this section, we prove that this function is not complete for  $\text{NP}_k^{\text{cc}}$  for  $k \geq 3$ .

For sets  $X_1, \dots, X_k$ , write  $\overline{X}$  for  $X_1 \times \dots \times X_k$ . We write  $\overline{x} \in \overline{X}$  to denote a  $k$ -tuple  $(x_1, \dots, x_k)$  where  $x_i \in X_i$  for all  $i \in [k]$ . Use  $\overline{\varphi}$  to denote a  $k$ -tuple of functions  $\varphi_1, \dots, \varphi_k$ . Furthermore, for  $i \in [k]$ , write  $\overline{\alpha}_{-i}$  for the  $(k - 1)$ -tuple obtained from  $\overline{\alpha}$  by removing the  $i$ -th coordinate.

In two-party communication complexity Babai, Frankl, and Simon [1] defined a natural notion of a reduction between problems called a ‘rectangular’ reduction that does not require any communication to compute.

**Definition 12.** For  $k = 2$ , let  $f : \overline{X} \rightarrow \{0, 1\}$  and  $g : \overline{X}' \rightarrow \{0, 1\}$ . A pair of functions  $\overline{\varphi}$  with  $\varphi_i : X_i \rightarrow X'_i$  is a rectangular reduction of  $f$  to  $g$ , written  $f \sqsubseteq g$ , if and only if  $f(x_1, x_2) = g(\varphi_1(x_1), \varphi_2(x_2))$ .

Furthermore, they defined an appropriate ‘polynomially-bounded’ version of rectangular reduction for function families.

**Definition 13.** For function families  $f = \{f_n\}$  and  $g = \{g_n\}$  where  $f_n, g_n : (\{0, 1\}^n)^2 \rightarrow \{0, 1\}$ , we write  $f \sqsubseteq_p g$  if and only if there is a function  $m : \mathbb{N} \rightarrow \mathbb{N}$  such that for every  $n$ ,  $f_n \sqsubseteq g_{m(n)}$  and  $m(n)$  is  $2^{(\log n)^{O(1)}}$ .

**Proposition 14 ([1]).** Let  $f$  and  $g$  be function families. If  $f \sqsubseteq_p g$  and  $g \in \text{P}_2^{\text{cc}}$  then  $f \in \text{P}_2^{\text{cc}}$ . If  $f \sqsubseteq_p g$  and  $g \in \text{NP}_2^{\text{cc}}$  then  $f \in \text{NP}_2^{\text{cc}}$ .

**Definition 15.** A function family  $g$  is complete for  $\text{NP}_2^{\text{cc}}$  under rectangular reductions if and only if  $g \in \text{NP}_2^{\text{cc}}$  and for all  $f \in \text{NP}_2^{\text{cc}}$ ,  $f \sqsubseteq_p g$ .

The set intersection function is  $\text{DISJ}_{k,n} : (\{0, 1\}^n)^k \rightarrow \{0, 1\}$  defined by  $\text{DISJ}_{k,n}(\overline{x}) = 1$  if and only if there is some  $i \in [n]$  such that  $x_{1,i} = \dots = x_{k,i} = 1$ . Clearly,  $\text{DISJ}_k \in \text{NP}_k^{\text{cc}}$ . Babai, Frankl and Simon observed the following:

**Proposition 16 ([1]).**  $\text{DISJ}_2$  is complete for  $\text{NP}_2^{\text{cc}}$  under rectangular reductions.



For  $k \geq 3$ , rectangular reductions extend to *cubic reductions* in the NIH model of communication complexity. Moreover, it is easy to see that the completeness result of Proposition 16 continues to hold in the NIH model under cubic reductions. One might conjecture that  $\text{DISJ}_k$  is also complete for  $\text{NP}_k^{cc}$  under a natural extension of rectangular reductions in the NOF model. Such a notion of reduction should not require any communication between the parties. This yields the following definition:

**Definition 17.** *Given  $f : \overline{X} \rightarrow \{0, 1\}$  and  $g : \overline{X'} \rightarrow \{0, 1\}$  we say that functions  $\overline{\varphi}$  are a cylindrical reduction of  $f$  to  $g$  if and only if for every  $\overline{x} \in \overline{X}$  there is an  $\overline{x'} \in \overline{X'}$  such that for all  $i \in [k]$ ,  $\varphi_i(\overline{x}_{-i}) = \overline{x'}_{-i}$  and  $f(\overline{x}) = g(\overline{x'})$ . Thus each  $\varphi_i$  maps the NOF view of the  $i$ -th player on input  $\overline{x}$  for  $f$  to the NOF view of the  $i$ -th player on input  $\overline{x'}$  for  $g$ .*

We show that cylindrical reductions must be of a special form, given by the natural no-communication reductions associated with the number-in-hand model.  $\overline{A} = A_1 \times \dots \times A_k$  is a *cube*, if  $A_i \subseteq X_i$  for all  $i \in [k]$ .

**Lemma 18.** *If there is a cylindrical reduction of  $f : \overline{X} \rightarrow \{0, 1\}$  to  $\text{DISJ}_{k,m}$  then  $f^{-1}(1)$  is a union of  $m$  cubes.  $\square$*

**Theorem 19.** *There is a function  $f : \{0, 1\}^{3n} \rightarrow \{0, 1\}$  with deterministic 3-party NOF communication complexity at most 3 such that any cylindrical reduction of  $f$  to  $\text{DISJ}_{3,m}$  requires  $m > 2^{n-3}$ .*

*Proof.* For  $x, y, z \in \{0, 1\}^n$ , define  $f(x, y, z)$  to be 1 if and only if  $x, y$ , and  $z$  are pairwise orthogonal in  $\mathbb{F}_2^n$ . There is a trivial 3-party NOF protocol for  $f$  in which 3 bits are exchanged, namely, each party checks that the inputs it sees are orthogonal. We now show that any way to write  $f^{-1}(1)$  as a union of cubes must contain exponentially many cubes since each cube can only cover an exponentially small portion of  $f^{-1}(1)$ .

For  $u, v \in \{0, 1\}^n$ , let  $h(u, v) = 1$  iff  $\langle x, y \rangle = 0$  in  $\mathbb{F}_2^n$ . Then  $f(x, y, z) = h(x, y)h(y, z)h(x, z)$ . Consider the uniform distribution  $\mu$  over  $\{0, 1\}^{3n}$ .

We first show that  $f^{-1}(1)$  is a set of probability more than 1/8. Under  $\mu$ , for each pair  $u, v \in \{x, y, z\}$ , the probability that  $h(u, v) = 1$  is  $1/2 + 1/2^{n+1} > 1/2$  (consider whether or not  $u = 0^n$ ). We claim that the probability that  $f(x, y, z) = 1$  is at least 1/8. Suppose that  $x \neq 0^n$ . Then the probability that  $y$  is orthogonal to  $x$  is precisely 1/2. Now,  $z$  is orthogonal to the span  $\langle \{x, y\} \rangle$  with probability at least 1/4. So, conditioned on  $x \neq 0^n$ , the probability that  $f(x, y, z) = 1$  is at least 1/8. If  $x = 0^n$  then the probability that  $f(x, y, z) = 1$  is precisely the probability that  $y$  and  $z$  are orthogonal which is at least 1/2. Therefore the probability that  $f(x, y, z) = 1$  is more than 1/8 overall.

Now since  $f(x, y, z) = h(x, y)h(y, z)h(x, z)$ , any cube  $C = A_1 \times A_2 \times A_3$  with  $C \subseteq f^{-1}(1)$  must, in particular, have,  $A_1 \times A_2 \subseteq h^{-1}(1)$ . Thus every  $x \in A_1$  must be orthogonal to every  $y \in A_2$  and so the dimensions of their spans must satisfy  $\dim(\langle A_1 \rangle) + \dim(\langle A_2 \rangle) \leq n$ . Therefore  $|A_1 \times A_2| \leq |\langle A_1 \rangle \times \langle A_2 \rangle| \leq 2^{\dim(\langle A_1 \rangle) + \dim(\langle A_2 \rangle)} \leq 2^n$  so  $|C| \leq 2^n |A_1 \times A_2| \leq 2^{2n}$  and the probability that  $(x, y, z) \in C$  is at most  $2^{-n}$ . The claimed result follows immediately.  $\square$

This argument can be extended to other functions  $h : \{0, 1\}^{2n} \rightarrow \{0, 1\}$  that have only small 1-monochromatic rectangles. It suffices that  $h(x, y)h(y, z)h(x, z)$  be 1 on a large fraction of inputs. Also, although the above Lemma is stated only for  $k = 3$  it is easy to see that the same bounds hold for larger  $k$ .

Given that any function  $f(x, y, z)$  of the form  $h_1(x, y)h_2(x, z)h_3(y, z)$  has communication complexity at most 3, it seems unlikely that any function is complete for  $\text{NP}_3^{\text{cc}}$  under efficient reductions that do not require communication.

## References

- [1] Babai, L., Frankl, P., Simon, J.: Complexity classes in communication complexity theory (preliminary version). In: FOCS27th, pp. 337–347 (1986)
- [2] Babai, L., Gál, A., Kimmel, P.G., Lokam, S.V.: Communication complexity of simultaneous messages. SJCOMP 33, 137–166 (2004)
- [3] Babai, L., Nisan, N., Szegedy, M.: Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. JCSS 45, 204–232 (1992)
- [4] Beame, P., Pitassi, T., Segerlind, N.: Lower bounds for lovász-schrijver systems and beyond follow from multiparty communication complexity. In: ICALP 32nd, pp. 1176–1188 (2005)
- [5] Beigel, R., Gasarch, W., Glenn, J.: The multiparty communication complexity of Exact-T: Improved bounds and new problems. In: MFCS31st, pp. 146–156 (2006)
- [6] Beigel, R., Tarui, J.: On ACC. In: FOCS 32nd, pp. 783–792 (1991)
- [7] Chandra, A.K., Furst, M.L., Lipton, R.J.: Multi-party protocols. In: STOC 15th, pp. 94–99 (1983)
- [8] Chung, F.R.K., Tetali, P.: Communication complexity and quasi randomness. SIAMDM 6, 110–125 (1993)
- [9] Ford, J., Gál, A.: Hadamard tensors and lower bounds on multiparty communication complexity. In: ICALP 32nd, pp. 1163–1175 (2005)
- [10] Håstad, J., Goldmann, M.: On the power of small-depth threshold circuits. CompCompl 1, 113–129 (1991)
- [11] Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press, Cambridge (1997)
- [12] Mansour, Y., Nisan, N., Tiwari, P.: The computational complexity of universal hashing. TCS 107, 121–133 (1993)
- [13] Newman, I.: Private vs. common random bits in communication complexity. IPL 39, 67–71 (1991)
- [14] Nisan, N.: The communication complexity of threshold gates. In: Proceedings of Combinatorics, Paul Erdos is Eighty, pp. 301–315 (1993)
- [15] Nisan, N., Wigderson, A.: Rounds in communication complexity revisited. SJCOMP 22, 211–219 (1993)
- [16] Raz, R.: The BNS-Chung criterion for multi-party communication complexity. CompCompl 9, 113–122 (2000)
- [17] Yao, A.C.C.: Some complexity questions related to distributive computing (preliminary report). In: STOC 11th, pp. 209–213 (1979)

# An Optimal Decomposition Algorithm for Tree Edit Distance

Erik D. Demaine, Shay Mozes\*, Benjamin Rossman, and Oren Weimann

MIT Computer Science and Artificial Intelligence Laboratory,  
32 Vassar Street, Cambridge, MA 02139, USA

edemaine@mit.edu, shaymozes@gmail.com, brossman@mit.edu, oweimann@mit.edu

**Abstract.** The *edit distance* between two ordered rooted trees with vertex labels is the minimum cost of transforming one tree into the other by a sequence of elementary operations consisting of deleting and re-labeling existing nodes, as well as inserting new nodes. In this paper, we present a worst-case  $O(n^3)$ -time algorithm for this problem, improving the previous best  $O(n^3 \log n)$ -time algorithm [7]. Our result requires a novel adaptive strategy for deciding how a dynamic program divides into subproblems, together with a deeper understanding of the previous algorithms for the problem. We prove the optimality of our algorithm among the family of *decomposition strategy* algorithms—which also includes the previous fastest algorithms—by tightening the known lower bound of  $\Omega(n^2 \log^2 n)$  [4] to  $\Omega(n^3)$ , matching our algorithm’s running time. Furthermore, we obtain matching upper and lower bounds of  $\Theta(nm^2(1 + \log \frac{n}{m}))$  when the two trees have sizes  $m$  and  $n$  where  $m < n$ .

## 1 Introduction

The problem of comparing trees occurs in diverse areas such as structured text databases like XML, computer vision, compiler optimization, natural language processing, and computational biology [2,3,8,11,13]. One major application is the analysis of RNA molecules in computational biology. The secondary structure of RNA, which plays a significant role in its biological function [9], is naturally represented as an ordered rooted tree [5,16]. Computing the similarity between the secondary structure of two RNA molecules therefore helps determine the functional similarities of these molecules.

The *tree edit distance* metric is a common similarity measure for rooted ordered trees. It was introduced by Tai in the late 1970’s [13] as a generalization of the well-known string edit distance problem [15]. Let  $F$  and  $G$  be two rooted trees with a left-to-right order among siblings and where each vertex is assigned a label from an alphabet  $\Sigma$ . The *edit distance* between  $F$  and  $G$  is the minimum cost of transforming  $F$  into  $G$  by a sequence of elementary operations consisting of deleting and relabeling existing nodes, as well as inserting new nodes (allowing at most one operation to be performed on each node). These operations are

---

\* Work conducted while visiting MIT.

illustrated in Fig. 1. The cost of elementary operations is given by two functions,  $c_{\text{del}}$  and  $c_{\text{match}}$ , where  $c_{\text{del}}(\tau)$  is the cost of deleting or inserting a vertex with label  $\tau$ , and  $c_{\text{match}}(\tau_1, \tau_2)$  is the cost of changing the label of a vertex from  $\tau_1$  to  $\tau_2$ . Since a deletion in  $F$  is equivalent to an insertion in  $G$  and vice versa, we can focus on finding the minimum cost of a sequence of just deletions and relabels in both trees that transform  $F$  and  $G$  into isomorphic trees.

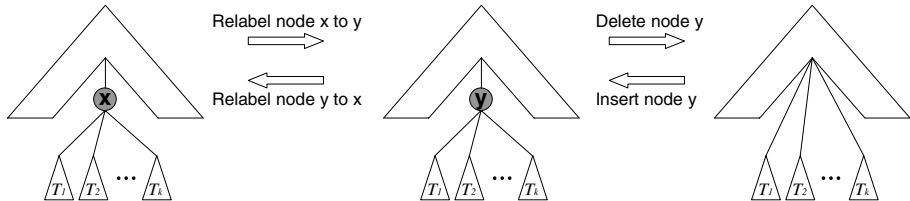


Fig. 1. The three editing operations on a tree with vertex labels

*Previous results.* To state running times, we need some basic notation. Let  $n$  and  $m$  denote the sizes  $|F|$  and  $|G|$  of the two input trees, ordered so that  $n \geq m$ . Let  $n_{\text{leaves}}$  and  $m_{\text{leaves}}$  denote the corresponding number of leaves in each tree, and let  $n_{\text{height}}$  and  $m_{\text{height}}$  denote the corresponding height of each tree, which can be as large as  $n$  and  $m$  respectively.

Tai [13] presented the first algorithm for computing tree edit distance, which requires  $O(n_{\text{leaves}}^2 m_{\text{leaves}}^2 nm)$  time and space, and thus has a worst-case running time of  $O(n^3 m^3) = O(n^6)$ . Shasha and Zhang [11] improved this result to an  $O(\min\{n_{\text{height}}, n_{\text{leaves}}\} \cdot \min\{m_{\text{height}}, m_{\text{leaves}}\} \cdot nm)$  time algorithm using  $O(nm)$  space. In the worst case, their algorithm runs in  $O(n^2 m^2) = O(n^4)$  time. Klein [7] improved this result to a worst-case  $O(m^2 n \log n) = O(n^3 \log n)$  time algorithm using  $O(nm)$  space. These last two algorithms are based on closely related dynamic programs, and both present different ways of computing only a subset of a larger dynamic program table; these entries are referred to as *relevant subproblems*. In [4], Dulucq and Touzet introduced the notion of a *decomposition strategy* (see Section 2.3) as a general framework for algorithms that use this type of dynamic program, and proved a lower bound of  $\Omega(nm \log n \log m)$  time for any such strategy. Many other solutions have been developed; see [1, 2, 14] for surveys. Among all these algorithms, Klein’s is the fastest in terms of worst-case time complexity, and previous improvements to Klein’s  $O(n^3 \log n)$  time bound were achieved only by constraining the edit operations or the scoring scheme [3, 10, 12, 17].

*Our results.* We present a new algorithm for computing the tree edit distance that falls into the same *decomposition strategy* framework of [4, 7, 11]. In the worst-case, our algorithm requires  $O(nm^2(1 + \log \frac{n}{m})) = O(n^3)$  time and  $O(nm)$  space. The corresponding sequence of edit operations can easily be obtained within the same time and space bounds. We therefore improve upon all known

algorithms in the worst-case time complexity. Furthermore, we prove a worst-case lower bound of  $\Omega(nm^2(1 + \log \frac{n}{m}))$  time for all decomposition strategy algorithms. This bound improves the previous best lower bound of  $\Omega(nm \log n \log m)$  time [4], and establishes the optimality of our algorithm among all decomposition strategy algorithms. Our algorithm is simple, making it easy to implement, but the analysis of the upper and lower bound proofs is quite complicated.

*Roadmap.* In Section 2 we give a simple and unified presentation of the two well-known tree edit algorithms, on which our algorithm is based, and on the class of decomposition strategy algorithms. We present and analyze the time complexity of our algorithm in Section 3, and prove the matching lower bound in Section 4. Final conclusions are presented in Section 5. For brevity, some of the proofs and an explicit  $O(nm)$  space complexity version of our algorithm will only be presented in the full version of this paper.

## 2 Background and Framework

Both the existing algorithms and ours compute the edit distance of finite ordered  $\Sigma$ -labeled forests, henceforth *forests*. The unique empty forest/tree is denoted by  $\emptyset$ . The vertex set of a forest  $F$  is written simply as  $F$ , as when we speak of a vertex  $v \in F$ . For a forest  $F$  and  $v \in F$ ,  $\sigma(v)$  denotes the  $\Sigma$ -label of  $v$ ,  $F_v$  denotes the subtree of  $F$  rooted at  $v$ , and  $F - v$  denotes the forest  $F$  after deleting  $v$ . The special case of  $F - \text{root}(F)$  where  $F$  is a tree is denoted  $F^\circ$ . The leftmost and rightmost trees of  $F$  are denoted by  $L_F$  and  $R_F$  and their roots by  $\ell_F$  and  $r_F$ . We denote by  $F - L_F$  the forest  $F$  after deleting the entire leftmost tree  $L_F$ ; similarly  $F - R_F$ . A forest obtained from  $F$  by a sequence of any number of deletions of the leftmost and rightmost roots is called a *subforest* of  $F$ .

Given forests  $F$  and  $G$  and vertices  $v \in F$  and  $w \in G$ , we write  $c_{\text{del}}(v)$  instead of  $c_{\text{del}}(\sigma(v))$  for the cost of deleting or inserting  $v$ , and we write  $c_{\text{match}}(v, w)$  instead of  $c_{\text{match}}(\sigma(v), \sigma(w))$  for the cost relabeling  $v$  to  $w$ .  $\delta(F, G)$  denotes the edit distance between the forests  $F$  and  $G$ .

Because insertion and deletion costs are the same (for a node of a given label), insertion in one forest is tantamount to deletion in the other forest. Therefore, the only edit operations we need to consider are relabels and deletions of nodes in both forests. In the next two sections, we briefly present the algorithms of Shasha and Zhang, and of Klein. This presentation, inspired by the tree similarity survey of Bille [2], is somewhat different from the original presentations and is essential for understanding our algorithm.

### 2.1 Shasha and Zhang's Algorithm [11]

Given two forests  $F$  and  $G$  of sizes  $n$  and  $m$  respectively, the following lemma is easy to verify. Intuitively, the lemma says that in any sequence of edit operations the two rightmost roots in  $F$  and  $G$  must either be matched with each other or else one of them is deleted.

**Lemma 1** ([11]).  $\delta(F, G)$  can be computed as follows:

- $\delta(\emptyset, \emptyset) = 0$
- $\delta(F, \emptyset) = \delta(F - r_F, \emptyset) + c_{\text{del}}(r_F)$
- $\delta(\emptyset, G) = \delta(\emptyset, G - r_G) + c_{\text{del}}(r_G)$
- $\delta(F, G) = \min \begin{cases} \delta(F - r_F, G) + c_{\text{del}}(r_F), \\ \delta(F, G - r_G) + c_{\text{del}}(r_G), \\ \delta(R_F^\circ, R_G^\circ) + \delta(F - R_F, G - R_G) + c_{\text{match}}(r_F, r_G) \end{cases}$

Lemma 1 yields an  $O(m^2n^2)$  dynamic program algorithm. If we index the vertices of the forests  $F$  and  $G$  according to their left-to-right postorder traversal position, then entries in the dynamic program table correspond to pairs  $(F', G')$  of subforests  $F'$  of  $F$  and  $G'$  of  $G$  where  $F'$  contains vertices  $\{i_1, i_1 + 1, \dots, j_1\}$  and  $G'$  contains vertices  $\{i_2, i_2 + 1, \dots, j_2\}$  for some  $1 \leq i_1 \leq j_1 \leq n$  and  $1 \leq i_2 \leq j_2 \leq m$ .

However, as we will presently see, only  $O(\min\{n_{\text{height}}, n_{\text{leaves}}\} \cdot \min\{m_{\text{height}}, m_{\text{leaves}}\} \cdot nm)$  different *relevant subproblems* are encountered by the recursion computing  $\delta(F, G)$ . We calculate the number of *relevant subforests* of  $F$  and  $G$  independently, where a forest  $F'$  (respectively  $G'$ ) is a relevant subforest of  $F$  (respectively  $G$ ) if it occurs in the computation of  $\delta(F, G)$ . Clearly, multiplying the number of relevant subforests of  $F$  and of  $G$  is an upper bound on the total number of relevant subproblems.

We now count the number of relevant subforests of  $F$ ; the count for  $G$  is similar. First, notice that for every node  $v \in F$ ,  $F_v^\circ$  is a relevant subproblem. This is because the recursion allows us to delete the rightmost root of  $F$  repeatedly until  $v$  becomes the rightmost root; we then match  $v$  (i.e., relabel it) and get the desired relevant subforest. A more general claim is stated and proved later on in Lemma 3. We define  $\text{keyroots}(F) = \{\text{the root of } F\} \cup \{v \in F \mid v \text{ has a left sibling}\}$ . It is easy to see that every relevant subforest of  $F$  is a prefix (with respect to the postorder indices) of  $F_v^\circ$  for some node  $v \in \text{keyroots}(F)$ . If we define  $\text{cdepth}(v)$  to be the number of keyroot ancestors of  $v$ , and  $\text{cdepth}(F)$  to be the maximum  $\text{cdepth}(v)$  over all nodes  $v \in F$ , we get that the total number of relevant subforest of  $F$  is at most

$$\sum_{v \in \text{keyroots}(F)} |F_v| = \sum_{v \in F} \text{cdepth}(v) \leq \sum_{v \in F} \text{cdepth}(F) = |F| \text{cdepth}(F).$$

This means that given two trees,  $F$  and  $G$ , of sizes  $n$  and  $m$  we can compute  $\delta(F, G)$  in  $O(\text{cdepth}(F)\text{cdepth}(G)nm) = O(n_{\text{height}}m_{\text{height}}nm)$  time. Shasha and Zhang also proved that for any tree  $T$  of size  $n$ ,  $\text{cdepth}(T) \leq \min\{n_{\text{height}}, n_{\text{leaves}}\}$ ; hence the result. In the worst case, this algorithm runs in  $O(m^2n^2) = O(n^4)$  time.

## 2.2 Klein’s Algorithm [7]

Klein’s algorithm is based on a recursion similar to Lemma 1. Again, we consider forests  $F$  and  $G$  of sizes  $|F| = n \geq |G| = m$ . Now, however, instead of recursing

always on the rightmost roots of  $F$  and  $G$ , we recurse on the leftmost roots if  $|L_F| \leq |R_F|$  and on the rightmost roots otherwise. In other words, the “direction” of the recursion is determined by the (initially) larger of the two forests. We assume the number of relevant subforests of  $G$  is  $O(m^2)$ ; we have already established that this is an upper bound.

We next show that Klein’s algorithm yields only  $O(n \log n)$  relevant subforests of  $F$ . The analysis is based on a technique called *heavy path decomposition* introduced by Harel and Tarjan [6]. Briefly: we mark the root of  $F$  as *light*. For each internal node  $v \in F$ , we pick one of  $v$ ’s children with maximal number of descendants and mark it as *heavy*, and we mark all the other children of  $v$  as *light*. We define  $\text{ldepth}(v)$  to be the number of light nodes that are ancestors of  $v$  in  $F$ , and  $\text{light}(F)$  as the set of all light nodes in  $F$ . By [6], for any forest  $F$  and vertex  $v \in F$ ,  $\text{ldepth}(v) \leq \log |F| + O(1)$ . Note that every relevant subforest of  $F$  is obtained by some  $i \leq |F_v|$  consecutive deletions from  $F_v$  for some light node  $v$ . Therefore, the total number of relevant subforests of  $F$  is at most

$$\sum_{v \in \text{light}(F)} |F_v| = \sum_{v \in F} \text{ldepth}(v) \leq \sum_{v \in F} (\log |F| + O(1)) = O(|F| \log |F|).$$

Thus, we get an  $O(m^2 n \log n) = O(n^3 \log n)$  algorithm for computing  $\delta(F, G)$ .

### 2.3 The Decomposition Strategy Framework

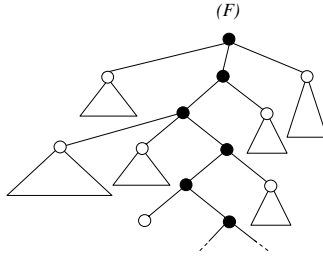
Both Klein’s and Shasha and Zhang’s algorithms are based on Lemma 1. The difference between them lies in the choice of when to recurse on the rightmost roots and when on the leftmost roots. The family of *decomposition strategy* algorithms based on this lemma was formalized by Dulucq and Touzet in [4].

**Definition 1 (Strategy, Decomposition Algorithm).** *Let  $F$  and  $G$  be two forests. A strategy is a mapping from pairs  $(F', G')$  of subforests of  $F$  and  $G$  to  $\{\text{left}, \text{right}\}$ . A decomposition algorithm is an algorithm based on Lemma 1 with the directions chosen according to a specific strategy.*

Each strategy is associated with a specific set of recursive calls (or a dynamic program algorithm). The strategy of Shasha and Zhang’s algorithm is  $S(F', G') = \text{right}$  for all  $F', G'$ . The strategy of Klein’s algorithm is  $S(F', G') = \text{left}$  if  $|L_{F'}| \leq |R_{F'}|$ , and  $S(F', G') = \text{right}$  otherwise. Notice that Shasha and Zhang’s strategy does not depend on the input trees, while Klein’s strategy depends only on the larger input tree. Dulucq and Touzet proved a lower bound of  $\Omega(mn \log m \log n)$  time for any decomposition strategy algorithm.

## 3 The Algorithm

In this section we present our algorithm for computing  $\delta(F, G)$  given two trees  $F$  and  $G$  of sizes  $|F| = n \geq |G| = m$ . The algorithm recursively uses a decomposition strategy in a divide-and-conquer manner to achieve  $O(nm^2(1 + \log \frac{n}{m})) = O(n^3)$



**Fig. 2.** A tree  $F$  with  $n$  nodes. Black nodes belong to the heavy path. White nodes are in  $\text{TopLight}(F)$ . The size of each subtree rooted at a white node is at most  $\frac{n}{2}$ .

running time in the worst case. For clarity we describe the algorithm recursively and analyze its time complexity. In the full version of this paper we prove that the space complexity can be made  $O(mn) = O(n^2)$ .

We begin with the observation that Klein’s strategy always determines the direction of the recursion according to the  $F$ -subforest, even in subproblems where the  $F$ -subforest is smaller than the  $G$ -subforest. However, it is not straightforward to change this since even if at some stage we decide to choose the direction according to the other forest, we must still make sure that all subproblems previously encountered are entirely solved. At first glance this seems like a real obstacle since apparently we only add new subproblems to those that are already computed. Our key observation is that there are certain subproblems for which it is worthwhile to choose the direction according to the *currently* larger forest, while for other subproblems we had better keep choosing the direction according to the *originally* larger forest.

For a tree  $F$  of size  $n$ , define the set  $\text{TopLight}(F)$  to be the set of roots of the forest obtained by removing the heavy path of  $F$  (i.e., the unique path starting from the root along heavy nodes). Note that  $\text{TopLight}(F)$  is the set of light nodes with  $\text{ldepth} = 1$  in  $F$  (see the definition of  $\text{ldepth}$  in section 2.2). This definition is illustrated in Fig. 2. Note that the following two conditions are always satisfied:

- (\*)  $\sum_{v \in \text{TopLight}(F)} |F_v| \leq n$ . Because  $F_v$  and  $F_{v'}$  are disjoint  $\forall v, v' \in \text{TopLight}(F)$ .
- (\*\*)  $|F_v| < \frac{n}{2}$  for every  $v \in \text{TopLight}(F)$ . Otherwise  $v$  would be a heavy node.

**THE ALGORITHM.** We compute  $\delta(F, G)$  recursively as follows:

- (1) If  $|F| < |G|$ , compute  $\delta(G, F)$  instead. That is, make  $F$  the larger forest.
- (2) Recursively compute  $\delta(F_v, G)$  for all  $v \in \text{TopLight}(F)$ . Along the way,  $\delta(F_{v'}, G_{w'})$  is computed and stored for all  $v'$  not in the heavy path of  $F$  and for all  $w \in G$ .



- (3) Compute  $\delta(F, G)$  using the following decomposition strategy:  $S(F', G') = \text{left}$  if  $F'$  is a tree, or if  $\ell_{F'}$  is not the heavy child of its parent. Otherwise,  $S(F', G') = \text{right}$ . However, do not recurse into subproblems that were previously computed in step (2).

The algorithm is evidently a decomposition strategy algorithm, since for all subproblems, it either deletes or matches the leftmost or rightmost roots. The correctness of the algorithm follows from the correctness of decomposition strategy algorithms in general.

*Time Complexity.* We show that our algorithm has a worst-case runtime of  $O(m^2n(1+\log \frac{n}{m})) = O(n^3)$ . We proceed by counting the number of subproblems computed in each step of the algorithm. Let  $R(F, G)$  denote the number of relevant subproblems encountered by the algorithm in the course of computing  $\delta(F, G)$ .

In step (2) we compute  $\delta(F_v, G)$  for all  $v \in \text{TopLight}(F)$ . Hence, the number of subproblems encountered in this step is  $\sum_{v \in \text{TopLight}(F)} R(F_v, G)$ . For step (3), we bound the number of relevant subproblems by multiplying the number of relevant subforests in  $F$  and in  $G$ . For  $G$ , we count all possible  $O(|G|^2)$  subforests obtained by left and right deletions. Note that for any node  $v'$  not in the heavy path of  $F$ , the subproblem obtained by matching  $v'$  with any node  $w$  in  $G$  was already computed in step (2). This is because any such  $v'$  is contained in  $F_v$  for some  $v \in \text{TopLight}(F)$ , so  $\delta(F_v^o, G_w^o)$  is computed in the course of computing  $\delta(F_v, G)$  (we prove this formally in Lemma 3). Furthermore, note that in step (3), a node  $v$  on the heavy path of  $F$  cannot be matched or deleted until the remaining subforest of  $F$  is precisely the tree  $F_v$ . At this point, both matching  $v$  or deleting  $v$  result in the same new relevant subforest  $F_v^o$ . This means that we do not have to consider matchings of nodes when counting the number of relevant subproblems in step (3). It suffices to consider only the  $|F|$  subforests obtained by deletions according to our strategy. Thus, the total number of new subproblems encountered in step (3) is bounded by  $|G|^2|F|$ .

We have established that if  $|F| \geq |G|$  then

$$R(F, G) \leq |G|^2|F| + \sum_{v \in \text{TopLight}(F)} R(F_v, G)$$

and if  $|F| < |G|$  then

$$R(F, G) \leq |F|^2|G| + \sum_{w \in \text{TopLight}(G)} R(F, G_w)$$

We first show, by a crude estimate, that this leads to an  $O(n^3)$  runtime. Later, we analyze the dependency on  $m$  and  $n$  accurately.

**Lemma 2.**  $R(F, G) \leq 4(|F||G|)^{3/2}$ .

*Proof.* We proceed by induction on  $|F| + |G|$ . The base of the induction is trivial. For the inductive step there are two symmetric cases. If  $|F| \geq |G|$  then  $R(F, G) \leq |G|^2|F| + \sum_{v \in \text{TopLight}(F)} R(F_v, G)$ . Hence, by the inductive assumption,

$$\begin{aligned} R(F, G) &\leq |G|^2|F| + \sum_{v \in \text{TopLight}(F)} 4(|F_v||G|)^{3/2} = |G|^2|F| + 4|G|^{3/2} \sum_{v \in \text{TopLight}(F)} |F_v|^{3/2} \\ &\leq |G|^2|F| + 4|G|^{3/2} \sum_{v \in \text{TopLight}(F)} |F_v| \max_{v \in \text{TopLight}(F)} \sqrt{|F_v|} \\ &\leq |G|^2|F| + 4|G|^{3/2}|F| \sqrt{\frac{|F|}{2}} = |G|^2|F| + \sqrt{8}(|F||G|)^{3/2} \leq 4(|F||G|)^{3/2} \end{aligned}$$

Here we have used facts (\*) and (\*\*) and the fact that  $|F| \geq |G|$ . The case where  $|F| < |G|$  is symmetric. □

This crude estimate gives a worst-case runtime of  $O(n^3)$ . We now analyze the dependence on  $m$  and  $n$  more accurately. Along the recursion defining the algorithm, we view step (2) as only making recursive calls, but not producing any relevant subproblems. Rather, every new relevant subproblem is created in step (3) for a unique recursive call of the algorithm. So when we count relevant subproblems, we sum the number of new relevant subproblems encountered in step (3) over all recursive calls to the algorithm. We define sets  $A, B \subseteq F$  as follows:

$$\begin{aligned} A &= \{a \in \text{light}(F) : |F_a| \geq m\} \\ B &= \{b \in F - A : b \in \text{TopLight}(F_a) \text{ for some } a \in A\} \end{aligned}$$

Note that the root of  $F$  belongs to  $A$ . Intuitively, the nodes in both  $A$  and  $B$  are exactly those for which recursive calls are made with the entire  $G$  tree. The nodes in  $B$  are the last ones, along the recursion, for which such recursive calls are made. We count separately:

- (i) the relevant subproblems created in just step (3) of recursive calls  $\delta(F_a, G)$  for all  $a \in A$ , and
- (ii) the relevant subproblems encountered in the entire computation of  $\delta(F_b, G)$  for all  $b \in B$  (i.e.,  $\sum_{b \in B} R(F_b, G)$ ).

Together, this counts all relevant subproblems for the original  $\delta(F, G)$ . To see this, consider the original call  $\delta(F, G)$ . Certainly, the root of  $F$  is in  $A$ . So all subproblems generated in step (3) of  $\delta(F, G)$  are counted in (i). Now consider the recursive calls made in step (2) of  $\delta(F, G)$ . These are precisely  $\delta(F_v, G)$  for  $v \in \text{TopLight}(F)$ . For each  $v \in \text{TopLight}(F)$ , notice that  $v$  is either in  $A$  or in  $B$ ; it is in  $A$  if  $|F_v| \geq m$ , and in  $B$  otherwise. If  $v$  is in  $B$ , then all subproblems arising in the entire computation of  $\delta(F_v, G)$  are counted in (ii). On the other hand, if  $v$  is in  $A$ , then we are in analogous situation with respect to  $\delta(F_v, G)$  as we were in when we considered  $\delta(F, G)$  (i.e., we count separately the subproblems created in step (3) of  $\delta(F_v, G)$  and the subproblems coming from  $\delta(F_u, G)$  for  $u \in \text{TopLight}(F_v)$ ).

Earlier in this section, we saw that the number of subproblems created in step (3) of  $\delta(F, G)$  is  $|G|^2|F|$ . In fact, for any  $a \in A$ , by the same argument, the

number of subproblems created in step (3) of  $\delta(F_a, G)$  is  $|G|^2|F_a|$ . Therefore, the total number of relevant subproblems of type (i) is  $|G|^2 \sum_{a \in A} |F_a|$ . For  $v \in F$ , define  $\text{depth}_A(v)$  to be the number of ancestors of  $v$  that lie in the set  $A$ . We claim that  $\text{depth}_A(v) \leq 1 + \log \frac{n}{m}$  for all  $v \in F$ . To see this, consider any sequence  $a_0, \dots, a_k$  in  $A$  where  $a_i$  is a descendent of  $a_{i-1}$  for all  $i \in [1, k]$ . Note that  $|F_{a_i}| \leq \frac{1}{2}|F_{a_{i-1}}|$  for all  $i \in [1, k]$  since the  $a_i$ s are light nodes. Also note that  $F_{a_0} \leq n$  and that  $|F_{a_k}| \geq m$  by the definition of  $A$ . It follows that  $k \leq \log \frac{n}{m}$ , i.e.,  $A$  contains no sequence of descendants of length  $> 1 + \log \frac{n}{m}$ . So clearly every  $v \in F$  has  $\text{depth}_A(v) \leq 1 + \log \frac{n}{m}$ .

We now have the number of relevant subproblems of type (i) as

$$|G|^2 \sum_{a \in A} |F_a| = m^2 \sum_{v \in F} \text{depth}_A(v) \leq m^2 \sum_{v \in F} (1 + \log \frac{n}{m}) = m^2 n (1 + \log \frac{n}{m}).$$

The relevant subproblems of type (ii) are counted by  $\sum_{b \in B} R(F_b, G)$ . Using Lemma 2, we have

$$\begin{aligned} \sum_{b \in B} R(F_b, G) &\leq 4|G|^{3/2} \sum_{b \in B} |F_b|^{3/2} \leq 4|G|^{3/2} \sum_{b \in B} |F_b| \max_{b \in B} \sqrt{|F_b|} \\ &\leq 4|G|^{3/2} |F| \sqrt{m} = 4m^2 n. \end{aligned}$$

Here we have used the facts that  $|F_b| < m$  and  $\sum_{b \in B} |F_b| \leq |F|$  (since the trees  $F_b$  are disjoint for different  $b \in B$ ). Therefore, the total number of relevant subproblems for  $\delta(F, G)$  is at most  $m^2 n (1 + \log \frac{n}{m}) + 4m^2 n = O(m^2 n (1 + \log \frac{n}{m}))$ . This implies:

**Theorem 1.** *The runtime of the algorithm is  $O(m^2 n (1 + \log \frac{n}{m}))$ . □*

## 4 A Tight Lower Bound for Decomposition Algorithms

In this section we present a lower bound on the worst-case runtime of decomposition strategy algorithms. We first give a simple proof of an  $\Omega(m^2 n)$  lower bound. In the case where  $m = \Theta(n)$ , this gives a lower bound of  $\Omega(n^3)$  which shows that our algorithm is worst-case optimal among all decomposition algorithms. To prove that our algorithm is worst-case optimal for any  $m \leq n$ , we analyze a more complicated scenario that gives a lower bound of  $\Omega(m^2 n (1 + \log \frac{n}{m}))$ , matching the running time of our algorithm, and improving the previous best lower bound of  $\Omega(nm \log n \log m)$  time [4].

In analyzing strategies we will use the notion of a *computational path*, which corresponds to a specific sequence of recursion calls. Recall that for all subforest-pairs  $(F', G')$ , the strategy  $S$  determines a direction: either right or left. The recursion can either delete from  $F'$  or from  $G'$  or match. A computational path is the sequence of operations taken according to the strategy in a specific sequence of recursive calls. For convenience, we sometimes describe a computational path by the sequence of subproblems it induces, and sometimes by the actual sequence of operations: either “delete from the  $F$ -subforest”, “delete from the  $G$ -subforest”, or “match”.

The following lemma states that every decomposition algorithm computes the edit distance between every two root-deleted subtrees of  $F$  and  $G$ .

**Lemma 3.** *Given a decomposition algorithm with strategy  $S$ , the pair  $(F_v^\circ, G_w^\circ)$  is a relevant subproblem for all  $v \in F$  and  $w \in G$  regardless of the strategy  $S$ .*

The proofs of Lemmas 3 and 4 are given in the full version of this paper. Lemma 4 establishes an  $\Omega(m^2n)$  lower bound on the number of relevant subproblems for any strategy.

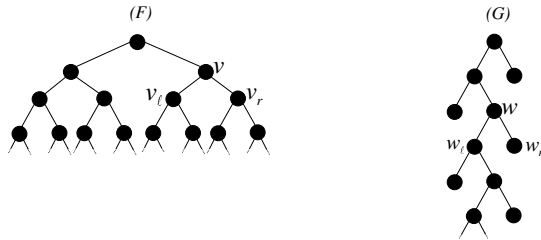
**Lemma 4.** *For any decomposition algorithm, there exists a pair of trees  $(F, G)$  with sizes  $n, m$  respectively, s.t. the number of relevant subproblems is  $\Omega(m^2n)$ .*

This lower bound is tight when  $m = \Theta(n)$ , since in this case our algorithm achieves an  $O(n^3)$  runtime. To establish a tight bound when  $m$  is not  $\Theta(n)$ , we use the following technique for counting relevant subproblems. We associate a subproblem consisting of subforests  $(F', G')$  with the unique pair of vertices  $(v, w)$  such that  $F_v, G_w$  are the smallest trees containing  $F', G'$  respectively. For example, for nodes  $v$  and  $w$  with at least two children, the subproblem  $(F_v^\circ, G_w^\circ)$  is associated with the pair  $(v, w)$ . Note that all subproblems encountered in a computational path starting from  $(F_v^\circ, G_w^\circ)$  until the point where either forest becomes a tree are also associated with  $(v, w)$ .

**Lemma 5.** *For every decomposition algorithm, there exists a pair of trees  $(F, G)$  with sizes  $n \geq m$  s.t. the number of relevant subproblems is  $\Omega(m^2n \log \frac{n}{m})$ .*

*Proof.* Consider the trees illustrated in Fig. 3. The  $n$ -sized tree  $F$  is a complete balanced binary tree, and  $G$  is a “zigzag” tree of size  $m$ . Let  $w$  be an internal node of  $G$  with a single node  $w_r$  as its right subtree and  $w_\ell$  as a left child. Denote  $m' = |G_w|$ . Let  $v$  be a node in  $F$  such that  $F_v$  is a tree of size  $n' + 1$  where  $n' \geq 4m \geq 4m'$ . Denote  $v$ 's left and right children  $v_\ell$  and  $v_r$  respectively. Note that  $|F_{v_\ell}| = |F_{v_r}| = \frac{n'}{2}$ .

Let  $S$  be the strategy of the decomposition algorithm. We aim to show that the total number of relevant subproblems associated with  $(v, w)$  or with  $(v, w_\ell)$  is at least  $\frac{n'}{4}(m' - 2)$ . Let  $c$  be the computational path that always deletes from  $F$  (no matter whether  $S$  says left or right). We consider two complementary cases.



**Fig. 3.** The two trees used to prove  $\Omega(m^2n \log \frac{n}{m})$  lower bound

CASE 1:  $\frac{n'}{4}$  left deletions occur in the computational path  $c$ , and at the time of the  $\frac{n'}{4}$ -th left deletion, there were fewer than  $\frac{n'}{4}$  right deletions.

We define a set of new computational paths  $\{c_j\}_{1 \leq j \leq \frac{n'}{4}}$  where  $c_j$  deletes from  $F$  up through the  $j$ th left deletion, and thereafter deletes from  $F$  whenever  $S$  says right and from  $G$  whenever  $S$  says left. At the time the  $j$ th left deletion occurs, at least  $\frac{n'}{4} \geq m' - 2$  nodes remain in  $F_{v_r}$  and all  $m' - 2$  nodes are present in  $G_{w_\ell}$ . So on the next  $m' - 2$  steps along  $c_j$ , neither of the subtrees  $F_{v_r}$  and  $G_{w_\ell}$  is totally deleted. Thus, we get  $m' - 2$  distinct relevant subproblems associated with  $(v, w)$ . Notice that in each of these subproblems, the subtree  $F_{v_\ell}$  is missing exactly  $j$  nodes. So we see that, for different values of  $j \in [1, \frac{n'}{4}]$ , we get disjoint sets of  $m' - 2$  relevant subproblems. Summing over all  $j$ , we get  $\frac{n'}{4}(m' - 2)$  distinct relevant subproblems associated with  $(v, w)$ .

CASE 2:  $\frac{n'}{4}$  right deletions occur in the computational path  $c$ , and at the time of the  $\frac{n'}{4}$ -th right deletion, there were fewer than  $\frac{n'}{4}$  left deletions.

We define a different set of computational paths  $\{\gamma_j\}_{1 \leq j \leq \frac{n'}{4}}$  where  $\gamma_j$  deletes from  $F$  up through the  $j$ th right deletion, and thereafter deletes from  $F$  whenever  $S$  says left and from  $G$  whenever  $S$  says right (i.e.,  $\gamma_j$  is  $c_j$  with the roles of left and right exchanged). Similarly as in case 1, for each  $j \in [1, \frac{n'}{4}]$  we get  $m' - 2$  distinct relevant subproblems in which  $F_{v_r}$  is missing exactly  $j$  nodes. All together, this gives  $\frac{n'}{4}(m' - 2)$  distinct subproblems. Note that since we never make left deletions from  $G$ , the left child of  $w_\ell$  is present in all of these subproblems. Hence, each subproblem is associated with either  $(v, w)$  or  $(v, w_\ell)$ .

In either case, we get  $\frac{n'}{4}(m' - 2)$  distinct relevant subproblems associated with  $(v, w)$  or  $(v, w_\ell)$ . To get a lower bound on the number of problems we sum over all pairs  $(v, w)$  with  $G_w$  being a tree whose right subtree is a single node, and  $|F_v| \geq 4m$ . There are  $\frac{m}{4}$  choices for  $w$  corresponding to tree sizes  $4j$  for  $j \in [1, \frac{m}{4}]$ . For  $v$ , we consider all nodes of  $F$  whose distance from a leaf is at least  $\log(4m)$ . For each such pair we count the subproblems associated with  $(v, w)$  and  $(v, w_\ell)$ . So the total number of relevant subproblems counted in this way is

$$\begin{aligned} \sum_{v,w} \frac{|F_v|}{4} (|G_w| - 2) &= \frac{1}{4} \sum_v |F_v| \sum_{j=1}^{\frac{m}{4}} (4j - 2) = \frac{1}{4} \sum_{i=\log 4m}^{\log n} \frac{n}{2^i} \cdot 2^i \sum_{j=1}^{\frac{m}{4}} (4j - 2) \\ &= \Omega(m^2 n \log \frac{n}{m}) \end{aligned} \quad \square$$

**Theorem 2.** For every decomposition algorithm and  $n \geq m$ , there exist trees  $F$  and  $G$  of sizes  $\Theta(n)$  and  $\Theta(m)$  s.t. the number of relevant subproblems is  $\Omega(m^2 n (1 + \log \frac{n}{m}))$ .

*Proof.* If  $m = \Theta(n)$  then this bound is  $\Omega(m^2 n)$  as shown in Lemma 4. Otherwise, this bound is  $\Omega(m^2 n \log \frac{n}{m})$  which was shown in Lemma 5. □

## 5 Conclusions

We presented a new  $O(n^3)$ -time and  $O(n^2)$ -space algorithm for computing the tree edit distance between two rooted ordered trees. Our algorithm is both symmetric in its two inputs as well as adaptively dependent on them. These features make it faster than all previous algorithms in the worst case. Furthermore, we proved that our algorithm is optimal within the broad class of decomposition strategy algorithms, by improving the previous lower bound for this class. Our algorithm is simple to describe and implement; our implementation in Python spans just a few dozen lines of code.

## References

1. Apostolico, A., Galil, Z. (eds.): Pattern matching algorithms. Oxford University Press, Oxford, UK (1997)
2. Bille, P.: A survey on tree edit distance and related problems. *Theoretical computer science* 337, 217–239 (2005)
3. Chawathe, S.S.: Comparing hierarchical data in external memory. In: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 90–101. Edinburgh, Scotland, U.K (1999)
4. Dulucq, S., Touzet, H.: Analysis of tree edit distance algorithms. In: Baeza-Yates, R.A., Chávez, E., Crochemore, M. (eds.) CPM 2003. LNCS, vol. 2676, pp. 83–95. Springer, Heidelberg (2003)
5. Gusfield, D.: Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge University Press, Cambridge (1997)
6. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. *SIAM Journal of Computing* 13(2), 338–355 (1984)
7. Klein, P.N.: Computing the edit-distance between unrooted ordered trees. In: Biliardi, G., Pietracaprina, A., Italiano, G.F., Pucci, G. (eds.) ESA 1998. LNCS, vol. 1461, pp. 91–102. Springer, Heidelberg (1998)
8. Klein, P.N., Tirthapura, S., Sharvit, D., Kimia, B.B.: A tree-edit-distance algorithm for comparing simple, closed shapes. In: Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 696–704 (2000)
9. Moore, P.B.: Structural motifs in RNA. *Annual review of biochemistry* 68, 287–300 (1999)
10. Selkow, S.M.: The tree-to-tree editing problem. *Information Processing Letters* 6(6), 184–186 (1977)
11. Shasha, D., Zhang, K.: Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing* 18(6), 1245–1262 (1989)
12. Shasha, D., Zhang, K.: Fast algorithms for the unit cost editing distance between trees. *Journal of Algorithms* 11(4), 581–621 (1990)
13. Tai, K.: The tree-to-tree correction problem. *Journal of the Association for Computing Machinery (JACM)* 26(3), 422–433 (1979)
14. Valiente, G.: Algorithms on Trees and Graphs. Springer, Heidelberg (2002)
15. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. *Journal of the ACM* 21(1), 168–173 (1974)
16. Waterman, M.S.: Introduction to computational biology: maps, sequences and genomes. chapters 13,14 Chapman and Hall (1995)
17. Zhang, K.: Algorithms for the constrained editing distance between ordered labeled trees and related problems. *Pattern Recognition* 28(3), 463–474 (1995)

# On Commutativity Based Edge Lean Search

Dragan Bošnački<sup>1</sup>, Edith Elkind<sup>2</sup>, Blaise Genest<sup>3</sup>, and Doron Peled<sup>4</sup>

<sup>1</sup> Department of Biomedical Engineering, Eindhoven University of Technology,  
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands

<sup>2</sup> Department of Computer Science, University of Southampton, U.K

<sup>3</sup> IRISA/CNRS, Campus de Beaulieu, 35042 Rennes Cedex, France

<sup>4</sup> Department of Computer Science, Bar Ilan University, Israel

**Abstract.** Exploring a graph through search is one of the most basic building blocks of various applications. In a setting with a huge state space, such as in testing and verification, optimizing the search may be crucial. We consider the problem of visiting all states in a graph where edges are generated by actions and the (reachable) states are not known in advance. Some of the actions may commute, i.e., they result in the same state for every order in which they are taken (this is the case when the actions are performed independently by different processes). We show how to use commutativity to achieve full coverage of the states while traversing considerably fewer edges.

## 1 Introduction

In many applications one has to explore a huge state space using limited resources (such as time and memory). Such applications include software and hardware testing and verification [4], multiagent systems, games (e.g., for the purpose of analyzing economic systems), etc. In such cases, it is obviously important to optimize the search, traversing only the necessary states and edges.

In this paper, we consider the problem of searching a large state space, where transitions between states correspond to a finite number of *actions*. We do not assume that the entire system is given to us as an input. Rather, we are given an initial state, and a method to generate states from one another. More specifically, for each state, there can be one or more actions available from this state. Executing an available action leads to another state. Also, we are given a fixed *independence* relation on actions: if two actions are independent, then executing them in any order from a given state leads to the same state. It is easy to model many of the problems in the above-mentioned application areas in this framework (we provide specific examples later in the paper). Traversing an edge and checking whether it leads to a new state has a cost. Hence, we want to predict if an edge is redundant (i.e., leads to a state that we have already visited or that we will necessarily visit in the future) without actually exploring it. Exploring fewer edges may also reduce the size of the search stack, and in particular reduce the memory consumption. Intuitively, the independence relation between actions should be useful here: if two sequences of actions lead to the same state,

it suffices to explore one of them. Of course, it will defy our goal to use a lot of overhead, both in terms of time and space spent computing the subset of edges to be explored.

The main contribution of this paper is an efficient state space search algorithm, which we call *commutativity-based edge-lean search* (CBEL). Our algorithm selects a total order on actions, extends it to paths (i.e., sequences of actions) in a natural way, and only explores paths that cannot be made smaller with respect to this order by permuting two adjacent independent actions. The proof that combining this simple principle with depth-first search ensures visiting all states turns out to be quite non-trivial (see Section 3). Another approach, which is inspired by trace theory [11,12], is to only consider paths that correspond to sequences in trace normal form (TNF) (defined later in the paper). This method provides a more powerful reduction than the one described above, as it only explores one sequence in each trace. In Section 4, we investigate this idea in more detail, describing an efficient data structure (called *summary*) that supports this search technique. We prove that the TNF-based algorithm is guaranteed to visit all states as long as the underlying state system contains no directed cycles. Unfortunately, if the state system is not loop-free, this algorithm may fail to reach some of the states: we provide an example in Section 4. While this limits the scope of applicability of this algorithm, many state systems, especially the ones that arise from game-theoretic applications, are naturally acyclic. Whenever this is the case, the TNF-based algorithm should be preferred over the algorithm of Section 3.

A related approach is the family of partial order reductions [13,5,14]. As opposed to our algorithms, in general these methods — known as *ample sets*, *persistent sets*, or *stubborn sets*, respectively, — do not preserve the property of visiting all the states, but guarantee to generate a reduced state space that preserves the property that one would like to check. Our algorithms are most closely related to the sleep set approach of [5], in particular, the non-state-splitting sleep set algorithms. In Section 5, we show that our TNF-based algorithm generates, in fact, exactly the same reduced graph as the very first version of the sleep set algorithm proposed in [6] (when edges are explored according to a fixed order between actions). The counterexample of Section 4 (see Figure 1) can therefore be seen as an explanation why in the presence of cycles, all existing sleep set algorithms have to use additional techniques to visit all states. In particular, the algorithm of [7] generates and checks back-edges, and discards them when redundant, thus paying the cost of checking some of the redundant edges. A fix suggested in [5,13] requires splitting nodes into several copies, which may increase the number of effective states. Surprisingly, our edge lean algorithm achieves a full coverage of the states without incurring these costs, and even in the presence of cycles.

An obvious application area for this technique is model checking and verification, but it can be useful in many other domains as well (see the full version of the paper [2]) To illustrate this, in Section 6 we describe an example from bioinformatics where one can use this method. In Section 7, we provide the results of



several experiments that compare our algorithms with classical depth-first search used in SPIN [9]. Our experiments show that for a number of natural problems, our methods provide a dramatic reduction in the number of edges explored and the stack size.

## 2 Preliminaries

A *system* is a tuple  $\mathcal{A} = \langle S, s_0, \Sigma, T \rangle$  such that

- $S$  is a finite set of *states*.
- $s_0 \in S$  is an *initial state*.
- $\Sigma$  is the finite *alphabet of actions*.
- $T \subseteq S \times \Sigma \times S$  is a labeled transition relation. We write  $s \xrightarrow{a} s'$  when  $(s, a, s') \in T$ .
- A symmetric and irreflexive relation  $I \subseteq \Sigma \times \Sigma$  on letters, called the *independence relation*. We require that independent transitions  $a I b$  satisfy the following *diamond* condition for every state  $s$ :

If  $s \xrightarrow{a} q \xrightarrow{b} r$  then there exists  $q' \in S$  such that  $s \xrightarrow{b} q' \xrightarrow{a} r$ . In this case, we say that the system has the *diamond* property.

Note that we do not require the other common diamond condition:

If  $s \xrightarrow{a} q$  and  $s \xrightarrow{b} q'$  then there exists  $r \in S$  such that  $s \xrightarrow{a} q \xrightarrow{b} r$ .

An action  $a$  is *enabled* from a state  $s \in S$  if there exists some state  $s' \in S$  such that  $s \xrightarrow{a} s'$ . We say that a path  $\rho = s_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} s_n$  is *loop-free* or *simple* if  $s_i \neq s_j$  for all  $i \neq j$ . Its *labeling* is  $\ell(\rho) = a_1 \dots a_n$ .

**Definition 1.** Let  $\sigma, \rho \in \Sigma^*$ . Define  $\sigma \stackrel{1}{\equiv} \rho$  iff  $\sigma = uabv$  and  $\rho = ubav$ , where  $u, v \in \Sigma^*$ , and  $a I b$ . Let  $\sigma \equiv \rho$  be the transitive closure of the relation  $\stackrel{1}{\equiv}$ . This equivalence relation is often called trace equivalence [11].

That is,  $\rho$  is obtained from  $\sigma$  (or vice versa) by commuting the order of an adjacent pair of letters. For example, for  $\Sigma = \{a, b\}$  and  $I = \{(a, b), (b, a)\}$  we have  $abbab \stackrel{1}{\equiv} ababb$  and  $abbab \equiv bbbaa$ . Notice that if the system has the diamond property and  $u \equiv v$ , then  $s \xrightarrow{u} r$  iff  $s \xrightarrow{v} r$ .

Let  $\ll$  be a total order on the alphabet  $\Sigma$ . We call it the *alphabetic order*. We extend  $\ll$  in the standard lexicographic way to words, i.e.,  $v \ll vu$  and  $vau \ll vbw$  for  $v, u, w \in \Sigma^*$ ,  $a, b \in \Sigma$  and  $a \ll b$ .

All the search algorithms to be presented are based on depth-first search (DFS), which provides a space complexity advantage over breadth-first search:

```

proc Dfs(q);
  local q';
  hash(q);
  forall q  $\xrightarrow{a}$  q' do
    if q' not hashed then Dfs(q');
end Dfs;
```

### 3 An Edge Lean Algorithm for Complete State Coverage

A key idea to reduce the number of explored edges is to make use of the diamond property, defined in the previous section.

**Definition 2.** Let  $w \in \Sigma^*$ . Denote by  $\tilde{w}$  the least word under the relation  $\ll$  equivalent to  $w$ . If  $w = \tilde{w}$ , then we say that  $w$  is in trace normal form (TNF) [12].

As  $\tilde{w} \equiv w$ , any state that can be reached by a path labeled with  $w$  can also be reached by a path labeled with  $\tilde{w}$ . Therefore, it is tempting to limit our attention to paths labeled with words in TNF, as such paths do explore *all* reachable states. However, one has to use caution when applying this approach within the depth-first search framework (see Section 4). The main reason for this is that all paths explored during depth-first search are necessarily acyclic. Hence, by using this method, we only consider paths that are *both* acyclic *and* labeled with words in TNF. On its own, neither of these restrictions prevents us from reaching all states. Unfortunately, it turns out that combining these limitations may result in leaving some states unexplored; we provide an example in Section 4. Therefore, for general state systems we have to settle for a less ambitious reduction. In what follows, we define a smaller relation on strings in  $\Sigma^*$ , and prove that it suffices to explore paths whose labeling is minimum with respect to this relation.

Set  $ubav \rightarrow_1 uabv$  if and only if  $aIb$  and  $a \ll b$  and let  $\rightarrow$  be the transitive closure of  $\rightarrow_1$ . We say that a word  $w \in \Sigma^*$  is *irreducible* if there exists no  $w' \neq w$  such that  $w \rightarrow w'$ . Intuitively, this means that  $w$  cannot be reduced, i.e., transformed into a smaller word with respect to  $\rightarrow$ , by a local fix (a single permutation of adjacent independent letters). We call a path  $\rho$  irreducible if its labeling  $\ell(\rho)$  is an irreducible word. Observe that a prefix of an irreducible path is also irreducible. Notice that if  $w$  is in TNF, then it is irreducible. The converse does not necessarily hold.

Our algorithm `EdgeLeanDfs` explores *some* irreducible paths in depth-first manner. For this, it suffices to remember the last letter  $a$  seen along the current path, and not to extend it with letter  $b$  whenever  $aIb$  and  $b \ll a$ .

```
EdgeLeanDfs( $s_0, \epsilon$ );
```

```
proc EdgeLeanDfs( $q, \text{prev}$ );
  local  $q'$ ;
  hash( $q$ );
  forall  $q \xrightarrow{a} q'$  such that  $\text{prev} = \epsilon$  or  $\neg(aI\text{prev})$  or  $\text{prev} \ll a$  do
    begin
      if  $q'$  not hashed then EdgeLeanDfs( $q', a$ );
    end
end EdgeLeanDfs;
```

Let `first_cbel( $s$ )` be the first path by which `EdgeLeanDfs( $s_0, \epsilon$ )` reaches the state  $s$ ; if `EdgeLeanDfs( $s_0, \epsilon$ )` does not reach  $s$ , set `first_cbel( $s$ )` =  $\perp$ .

**Theorem 1.** For any  $s \in \mathcal{A}$ , we have `first_cbel( $s$ )`  $\neq \perp$ , i.e., our algorithm explores all states. This implies that `EdgeLeanDfs( $s_0, \epsilon$ )` is correct.

*Proof.* To prove Theorem III, we fix a state  $s$ , and show that  $\text{EdgeLeanDfs}(s_0, \epsilon)$  reaches this state. To do so, we start with an arbitrary simple irreducible path in the state graph that reaches  $s$  (we show that such path always exists) and repeatedly apply to it a transformation  $T$ , defined below. This transformation produces another simple irreducible path that also leads to  $s$ . We show that for any  $\rho$  for which  $T(\rho)$  is defined, an application of  $T$  results in a path that is smaller than  $\rho$  with respect to a certain well-founded ordering, defined later. Therefore, after a finite number of iterations, we obtain a simple irreducible path  $\rho$  such that  $T(\rho)$  is not defined. We then prove that any such  $\rho$  is a path taken by  $\text{EdgeLeanDfs}(s_0, \epsilon)$ , i.e.,  $s$  is reached by our algorithm. The details follow.

For any simple path  $\rho$  and any state  $t$  on this path, we denote by  $\rho_t$  the prefix of  $\rho$  that reaches  $t$ ; in particular,  $\rho_s$  is a simple path that reaches  $s$ . We will now show that we can choose  $\rho_s$  so that it is irreducible.

**Claim 1.** *For any path  $\rho_s$ , there exists a path  $\rho'_s$  that is simple and irreducible.*

*Proof.* We iteratively (1) delete loops and (2) rearrange the labels to obtain an irreducible path. Each application of (1) strictly decreases the length of the path, while (2) does not change its length. The path obtained also leads to  $s$ . We obtain a simple irreducible path after a finite number of iterations. ■

Given a simple path  $\rho$  that reaches  $s$ , all states on  $\rho$  can be classified into three categories with respect to  $\rho$ : we say that a state  $t$  is *red* if  $\text{first\_cbel}(t) = \rho_t$ , *blue* if  $\text{first\_cbel}(t) \neq \perp$ , but  $\text{first\_cbel}(t) \neq \rho_t$ , and *white* if  $\text{first\_cbel}(t) = \perp$ . This classification depends on the path  $\rho$ : a state can be red with respect to one path but blue with respect to a different path. It turns out that for a simple irreducible path, not all sequences of state colors are possible.

**Lemma 1.** *Suppose that  $\rho_s$  is loop-free and irreducible. Then if  $t$  is the last red state along  $\rho_s$ , all states that precede  $t$  on  $\rho_s$  are also red. Moreover, either  $t$  is the last state on  $\rho_s$ , i.e.,  $t = s$ , or the state  $t'$  that follows  $t$  on  $\rho_s$  is blue.*

*Proof.* The first statement of the lemma follows from the definition of a red state and from the use of depth-first search. To prove the second statement, assume for the sake of contradiction that  $t'$  is white ( $t'$  cannot be red as  $t$  is the last red state on  $\rho_s$ ). The path  $\rho_{t'}$  is a prefix of  $\rho_s$ , so it is simple and irreducible. Hence,  $\text{EdgeLeanDfs}(s_0, \epsilon)$  must explore the transition that leads from  $t$  to  $t'$ . Therefore,  $t'$  cannot be white. ■

We define a transformation  $T$  that can be applied to any simple irreducible path  $\rho = \rho_s$  that contains a blue state; its output is another simple irreducible path that reaches the same state  $s$ . Recall that  $\ell(\pi)$  denotes the labeling of a path  $\pi$ . The transformation  $T$  consists of the following steps ( $w$  and  $v$  appear only for a later reference in the proof):

- (1) Let  $\rho_t$  be the shortest prefix of  $\rho$  such that  $t$  is blue. Decompose  $\rho$  as  $\rho = \rho_t \sigma$ . Modify  $\rho$  by replacing  $\rho_t$  with  $\text{first\_cbel}(t)$ , i.e., set  $\rho = \text{first\_cbel}(t)\sigma$ . Set  $w = y = \ell(\text{first\_cbel}(t))$ ,  $v = z = \ell(\sigma)$  and  $x = yz = \ell(\rho)$ .

- (2) Eliminate all loops from  $\rho$ . Update  $x$ ,  $y$ , and  $z$  by deleting the substrings that correspond to these loops.
- (3) Replace  $\rho$  with an equivalent irreducible path as follows.
  - (3a) Replace  $z$  with an equivalent irreducible word.
  - (3b) Let  $a$  be the last letter of  $y$ , and let  $b$  be the first letter of  $z$ . If  $a \gg b$  and  $aIb$ , move  $a$  from  $y$  to  $z$  and push it as far to the right as possible within  $z$ .
  - (3c) Repeat Step (3b) until the last letter  $a$  of  $y$  cannot be moved to  $z$ , i.e.,  $a \ll b$  or  $a$  and  $b$  are not independent.
  - (3d) Set  $x = yz$ , and let  $\rho$  be a path reaching  $s$  with  $\ell(\rho) = x$ .
- (4) Repeat (2) and (3) until  $\rho$  is simple and irreducible.

By the argument in the proof of Claim 1, we only need to repeat Steps (2) and (3) a finite number of times, so the computation of  $T$  terminates. Observe that if  $s$  is red with respect to  $\rho_s$ , then  $T(\rho_s)$  is not defined. On the other hand, consider a simple irreducible path  $\rho_s$  such that  $s$  is not red with respect to  $\rho_s$ . By Lemma 1, we can apply  $T$  to  $\rho_s$ . The output of  $T(\rho_s)$  is loop-free and irreducible, so if  $s$  is not red with respect to  $T(\rho_s)$ , we can apply  $T$  to  $T(\rho_s)$ . We will now show that after a finite number of iterations  $n$ , we obtain a path  $T^n(\rho_s)$ , which consists of red states only.

**Definition 3.** For a word  $v \in \Sigma^*$ , let  $\#_a(v)$  be the number of occurrences of the letter  $a$  in  $v$ . We write  $v <_{\#} w$  if there exists a letter  $a$  such that for all  $b \ll a$ ,  $\#_b(v) = \#_b(w)$  and  $\#_a(v) < \#_a(w)$ .

**Claim 2.** The relation  $<_{\#}$  is a well-founded (partial) order, i.e., there does not exist an infinite sequence  $u_1, u_2, \dots, u_i \in \Sigma^*$  such that  $u_1 >_{\#} u_2 >_{\#} \dots$ .

Consider a simple irreducible path  $\rho = \rho_s$ . Suppose that both  $\rho$  and  $T(\rho)$  contain blue states.

**Lemma 2.** Let  $\rho = \rho_t \sigma$ , where  $t$  is the first blue state on  $\rho$ , and let  $T(\rho) = \rho'_t \sigma'$ , where  $t'$  is the first blue state on  $T(\rho)$ . Let  $v = \ell(\sigma)$ ,  $v' = \ell(\sigma')$ . Then  $v >_{\#} v'$ .

Before we prove the lemma, let us show that it implies Theorem 1. Indeed, by Claim 2, there does not exist an infinite decreasing sequence of words with respect to  $<_{\#}$ . The strings  $v, v'$  satisfy  $v' <_{\#} v$ , and are well-defined as long as both  $\rho$  and  $T(\rho)$  contain blue states. Hence, for some finite value of  $n$ ,  $T^n(\rho)$  contains no blue states, it is simple and irreducible. Therefore, by Lemma 1 we obtain a path of our algorithm that reaches  $s$ . We now prove Lemma 2.

*Proof.* We use the notation introduced in the description of  $T$ : we have  $w = \ell(\text{first\_cbel}(t))$ ,  $v = \ell(\sigma)$ , and  $x = wv = \ell(\rho)$  after  $\rho_t$  was replaced by  $\text{first\_cbel}(t)$ .

In the rest of the proof, we abuse notation by using the word ‘letter’ to refer both to an element of  $\Sigma$  and an occurrence of this element in a word. The specific meaning will be clear from the context. In particular, we will assign colors to occurrences of the elements of  $\Sigma$  rather than the elements itself, whereas when we write  $a \ll b$ , we refer to the respective elements of  $\Sigma$ .

Let us color all the letters in the word  $wv$  so that all letters in  $w$  are yellow and all letters in  $v$  are green. By construction at any point in time all letters in  $y$  are yellow, and therefore all letters pushed into  $z$  during Step (3) are yellow. We construct a directed acyclic graph (DAG) whose set of nodes includes all yellow occurrences of letters in  $z$  as well as some of the green occurrences of letters. Namely, if a yellow letter  $a$  gets pushed into  $z$  when the first letter of  $z$  is  $b$ , there is an edge from this occurrence of  $a$  to this occurrence of  $b$ . Also, if a (yellow or green) letter  $a$  that is currently the first letter of  $z$  gets transposed with its right-hand side neighbor  $b$  (by (3a)), there is an edge from this occurrence of  $a$  to this occurrence of  $b$ . Observe that in both cases if there is an edge from an occurrence of  $a$  to an occurrence of  $b$ , then we have  $b \ll a$ , so our graph contains no directed cycles. We do not delete a node from this graph even if the respective occurrence is deleted from  $x$  by (2).

**Claim 3.** *Each yellow letter pushed into  $z$  has an outgoing edge. Moreover, if a letter has incoming edges, but no outgoing edges, either it has been eliminated from  $x$ , or it is the first letter of  $z$  after the execution of  $T$  is completed.*

*Proof.* Each yellow letter acquires an outgoing edge as it is moved into  $z$ . Now, consider a letter that has incoming edges. It acquired them either when it was the first letter of  $z$  and yellow letters were pushed past it, or when it was transposed with its left-hand side neighbor and became the first letter of  $z$ . In both cases, it was the first letter of  $z$  at some point in time. If it remains in that position till the end of the execution of  $T$ , we are done. Now, suppose that it stopped being the first letter of  $z$ . Then either it was deleted during loop elimination phase, in which case we are done, or it was transposed with its right-hand side neighbor, in which case it acquired an outgoing edge. ■[Claim 3]

Let  $G$  be the set of nodes of our DAG that have incoming edges, but no outgoing edges. By Claim 3 none of the letters in  $G$  is yellow, so all of them are green. Moreover, each letter in  $G$  either has been eliminated from  $x$  or is the first letter of  $z$  after the end of the execution of  $T$ .

Consider the string  $x = yz$  obtained after the end of the execution of  $T$ . This string corresponds to  $\rho' = T(\rho)$ . Recall that  $w$  corresponds to  $\text{first\_cbel}(t)$ , which consists of red states only, and  $y$  is a prefix of  $w$ . Hence, the prefix of  $\rho'$  that corresponds to  $y$  reaches a red state. Therefore, to reach a blue state along  $\rho'$ , we need to progress over at least one letter of  $z$ , or, equivalently,  $v'$  is a strict suffix of  $z$ , that is,  $v'$  does not include the first letter of  $z$ . Using Claim 3, we conclude that  $v'$  does not contain any of the letters in  $G$ .

Let  $a$  be the minimal (for  $\ll$ ) letter of  $G$ . It is contained in  $v$ , but not in  $v'$ . On the other hand, each letter  $c$  that is contained in  $v'$ , but not in  $v$ , is a yellow letter that appears in the DAG, that is there is a path of the DAG leading from  $c$  to some  $b \in G$ . By construction of the graph, the existence of a path from  $c$  to  $b$  implies  $c \gg b$ , and hence  $c \gg a$ . Hence, for all  $b$  in  $v'$  with  $b \ll a$  or  $b = a$ ,  $b$  is green, hence  $\#_b(v') \leq \#_b(v)$ , and  $a \in G$  is in  $v$  but not in  $v'$ , hence  $\#_a(v') < \#_a(v)$ , that is  $v' <_{\#} v$ . ■[Lemma 2, Theorem 1]

## 4 An Efficient Reduction for Cycle Free State Spaces

It can be argued that the reduction of Section 3 is not optimal: let  $a \ll b \ll c$ ,  $aIb, bIc$  and  $\neg(aIc)$ . Let  $x = cab$  and  $y = bca$ . Then we have  $x \equiv y$ , i.e., the states reached after  $x$  and  $y$  are the same. However, both  $x$  and  $y$  are irreducible, since  $a \ll b$  and  $\neg(aIc)$ . Therefore, the algorithm of Section 3 will explore both of the paths labeled by  $x$  and  $y$ .

In this section, we describe an algorithm `TNF_Dfs`( $s_0$ ) that only explores paths labeled with words in trace normal form. This algorithm often provides a significant reduction in the size of stack needed, both compared to `DFS` and `EdgeLeanDfs`. For acyclic state spaces, `TNF_Dfs`( $s_0$ ) explores all states. However, it may not be the case in general. In the end of this section, we provide an example in which some of the states are not reached. Denote by  $\alpha(\sigma)$  the set of letters occurring in  $\sigma$ .

**Definition 4.** A summary of a string  $\sigma$  is the total order  $\prec_\sigma$  on the letters from  $\alpha(\sigma)$  such that  $a \prec_\sigma b$  iff the last occurrence of  $a$  in  $\sigma$  precedes the last occurrence of  $b$  in  $\sigma$ . That is,  $\sigma = vauwb$ , where  $v \in \Sigma^*$ ,  $u \in (\Sigma \setminus \{a\})^*$ ,  $w \in (\Sigma \setminus \{a, b\})^*$ .

Our reduction will be based on generating paths that are in TNF. The proofs of the following two lemmas can be found in the full version of the paper [2].

**Lemma 3.** Let  $\sigma \in \Sigma^*$  be in TNF and  $a \in \Sigma$ . Then  $\sigma a$  is not in TNF if and only if  $\sigma = vu$  for some  $v, u$  such that (i)  $vau \equiv vua$  and (ii)  $vau \ll vu$ .

Intuitively, Lemma 3 means that we can commute the last  $a$  in  $vua$  backwards over  $u$  to obtain a string that is smaller in the alphabetic order than  $vu$ . The following lemma shows how we can use a summary to decide whether  $\sigma a$  is in TNF. It implies that it suffices to consider the suffix of the summary that commutes with  $a$ , and look among these letters for one that comes after  $a$  in the alphabetic order.

**Lemma 4.** Let  $\sigma \in \Sigma^*$  in TNF and  $a \in \Sigma$ . Then  $\sigma a$  is not in TNF if and only if there is a  $b \in \alpha(\sigma)$  such that  $a \ll b$  and for each  $c$  such that  $b \preceq_\sigma c$ ,  $aIc$ .

To perform a reduced depth-first search (DFS) that only considers strings in TNF, we store the summary in a global array `summary[1..n]`, where  $n = |\Sigma|$ . The variable `size` stores the number of different letters in the current string  $\sigma$ . We update the summary as we progress with the DFS, and recover the previous value when backtracking, i.e., the value of the summary is calculated on the fly and not stored with the state information.

The reduced DFS procedure `TNF_Dfs`( $s_0$ ) considers all transitions enabled at the current state. For each of them, it checks whether the current string augmented with this transition is in TNF. This is done through a call to the function `normal`, which checks the summary, according to Lemma 4. The pseudocode description of auxiliary functions used by `TNF_Dfs` can be found in [2].

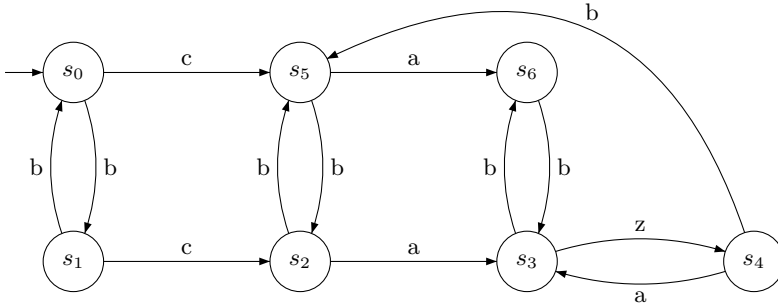


Fig. 1. A state space for which TNF\_DFS does not explore every state

```

proc TNF_Dfs(q);
  local q', i;
  hash(q);
  forall q  $\xrightarrow{a}$  q' in increasing  $\ll$ -order do
    if normal(a) and q' not hashed then
      i:=ord(a);
      update_sumr(i,a);
      TNF_Dfs(q');
      recover_sumr(i,a);
  end TNF_Dfs;

```

**Theorem 2.** Given an acyclic state space  $\mathcal{A}$ , the algorithm  $\text{TNF\_Dfs}(s_0)$  visits all states of  $\mathcal{A}$ .

*Proof.* We show that every state  $s$  is reached by the path  $\text{first}(s)$ , where  $\text{first}(s)$  stands for the path labeled by the minimal (for  $\ll$ ) word reaching  $s$ . Clearly,  $\text{first}(s)$  is in TNF. By contradiction, if it is not the case, take the state with the smallest  $\text{first}(s)$  such that  $s$  is not explored by  $\text{first}(s)$ . Then  $\text{first}(s) = ua$ , with  $u$  reaching  $t$  with  $u \ll \text{first}(s)$ , hence  $u = \text{first}(t)$ . When considering  $a$ ,  $ua$  is in TNF and *acyclic*, hence  $s$  will be reached by  $ua = \text{first}(s)$ , and since  $\text{first}(s)$  is minimal for  $\ll$ , no other path that reaches  $s$  has been considered before. ■

Unfortunately, for graphs that contain cycles, the conclusion of Theorem 2 is no longer true since  $ua$  and the minimal word reaching  $s$  may have loops. Figure 1 provides an example of a (diamond closed) graph that is not fully covered by the TNF algorithm (and hence, as shown in the next section, neither by the SleepSetsDfs version of the sleep set algorithm). The nodes, except  $s_6$ , are ordered in the order in which they are discovered. The node  $s_6$  is not discovered. The alphabet is  $\{a, b, c, z\}$ , with the ordering  $a \ll b \ll c \ll z$ . The independence relation is given by  $bIa, bIc$ . Consequently,  $z$  depends on every other letter  $a, b, c$ , and  $a, c$  are dependent. The state  $s_6$  can only be visited through

$s_3$  and  $s_5$ , with  $\text{first}(s_3) = bca$  and  $\text{first}(s_5) = bcazb$ . Neither  $bcab$  ( $\equiv bbca$ ) nor  $bcazba$  ( $\equiv bcaaab$ ) is in TNF (but note that  $bcab$  is irreducible, as required by `EdgeLeanDfs`), hence  $s_6$  is not visited. On the other hand,  $s_6$  is visited by `EdgeLeanDfs`.

## 5 TNF\_Dfs and Sleep Sets

In the full version of the paper, we explore the relationship between the algorithms of Sections 3 and 4, and the sleep set approach 5. Here, we show that the most straightforward version of the sleep set approach 6 is equivalent to `TNF_Dfs`.

```

proc SleepSetsDfs(s,sleep);
  local s', current;
  current:=sleep;
  hash(s);
  forall a ∉ sleep, s  $\xrightarrow{a}$  s' do
    begin
      if s' not hashed then
        SleepSetDfs(s',current \{b | ¬(b I a)});
        current := current ∪ {a};
      end;
    end
end SleepSetsDfs;

```

**Lemma 5.** *Assume that both `TNF_Dfs` and `SleepSetsDfs` use the same alphabetic priority order  $\ll$ . Then from any given state  $s$  during the search they explore exactly the same successors.*

*Proof.* Consider an action  $a$  that is in the sleep set of a state  $s$ . Suppose that  $s$  is reachable from the initial state via a path labeled with  $\sigma$  in TNF. Then  $\sigma$  can be decomposed as  $\sigma = vu$  so that there is a state  $t$  reached from  $s_0$  via  $v$ ,  $a$  has been taken from  $t$ , and all the letters in  $u$  are independent of  $a$ . According to the priority  $\ll$ , we have  $a \ll u$ . Thus, if  $a$  is in the sleep set of  $s$ , according to Lemma 3,  $\sigma a$  cannot be in normal form.

Conversely, assume that `TNF_Dfs` does not take a transition labeled with  $a$  after a state  $s$ , where the path on the stack is labeled with  $\sigma$ . This is because  $\sigma a$  is not in normal form. As per Lemma 3, let  $u$  be the longest suffix of  $\sigma$  such that  $\sigma = vu$ ,  $vau \equiv vua$  and  $vau \ll vu$ . Let  $t$  be the state reached after  $v$ . Then  $a$  is enabled from  $t$ . Now, consider the sleep set algorithm. If  $a$  is taken from  $t$ , it will be taken before the first letter of  $u$ , according to the lexicographic order priority (since  $a \ll u$ ). Then  $a$  must be in the sleep set of  $s$ , and thus is not taken from it. Since  $a$  is enabled at  $t$ , if  $a$  is not taken from  $t$ , it must be because  $a$  is in the sleep set when we reach  $t$ . But this means that there is a longer suffix  $u'$  of  $\sigma$  such that  $a$  is independent of  $u'$ , and  $a \ll u'$ , a contradiction to the maximality of  $u$ . ■



## 6 Applications

The idea of speeding up state-space search by using independence relation between actions is well-known in the model-checking community. In this section, we present an example motivated by bioinformatics, where one can also apply this technique. In [2], we provide examples from other domains such as voting theory, cellular automata theory and auction theory.

**Mining Maximal Frequent Itemsets.** Mining maximal itemsets is an important problem in data mining (e.g. [8]) with various applications in other areas like, for instance, bioinformatics [10]. We assume a set  $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$  of  $m$  distinct items and a database of  $n$  transactions  $\mathcal{D} = t_1, t_2, \dots, t_n$ . Each transaction is a subset of  $\mathcal{I}$ . Let  $X \subseteq \mathcal{I}$ . We define the *support*  $\sigma(X)$  of  $X$  as the number of transactions in which  $X$  occurs as a subset. A set is *frequent* iff  $\sigma(X) \geq \min_{sup}$ , with some *minimum support value*  $\min_{sup}$ . A frequent set is *maximal* if it is not a subset of any other frequent set. The algorithm for generating all maximal frequent sets [8] is a backtracking algorithm that, beginning with an empty set, builds frequent sets by adding one item at a time. An item is added to the current (frequent) set only if the new set that is obtained is frequent too. The algorithm generates a state space whose states are frequent sets. The transitions correspond to the actions of adding a new item to the set. The choice which item will be added to the set is obviously non-deterministic. Two actions are independent if they add items that are contained in all sets of the database. The maximal frequent sets correspond to “deadlock” states, i.e., states (sets) that cannot be further extended. Since added items are never removed from a set the state space graph is acyclic. A variant of this algorithm is applied in bioinformatics for finding similarities between biological networks [10]. For this purpose the original problem is transformed into a one of finding maximal subgraphs in a collection of undirected graphs. The graphs are represented as sets of edges, therefore there is a one-to-one correspondence with the original problem (edges, graphs, collection of graphs, vs. items, sets, database, respectively).

## 7 Experiments

We implemented the algorithms `EdgeLeanDfs` and `TNF_Dfs` from Sections 3 and 4 in the tool Spin [9]. We tested state space generation of examples from the literature. The results are shown in Table 1. The columns correspond to regular depth-first search, the edge lean algorithm, and the TNF-based algorithm, respectively. For each example we give the number of states and edges explored, and the maximal size of the stack, in unit, thousands (K) and millions (M). The first two examples, `DMSnoCC` and `DMSwithCC`, are models of system-on-chip designs of a distributed memory system on message passing network without and with cache coherency, respectively ([1]). The examples `RW1`, `RW4`, and `RW6` are models of various instances of the so-called Replicated Workers problem described in [3]. The rest of the models are from the test suite that comes

**Table 1.** Experimental Results

| model     | Spin with regular DFS |       |       | Spin with EdgeLeanDfs |       |       | Spin with TNF_Dfs |       |       |
|-----------|-----------------------|-------|-------|-----------------------|-------|-------|-------------------|-------|-------|
|           | states                | edges | stack | states                | edges | stack | states            | edges | stack |
| DMSnoCC   | 229M                  | 1009M | 26M   | 229M                  | 296M  | 47,3K | 229M              | 265M  | 32,2K |
| DMSwithCC | 132M                  | 541M  | 18,9M | 132M                  | 174M  | 384K  | 132M              | 151M  | 29,2K |
| RW1       | 181K                  | 852K  | 2219  | 181K                  | 409K  | 1224  | 181K              | 339K  | 360   |
| RW4       | 263K                  | 1.1M  | 2253  | 263K                  | 558K  | 1247  | 263K              | 462K  | 625   |
| RW6       | 11.5M                 | 65.6M | 827K  | 11.5M                 | 59.6M | 784K  | 9.9M              | 41.3M | 148K  |
| petersonN | 25362                 | 69787 | 5837  | 25362                 | 28855 | 1035  | 25362             | 28328 | 632   |
| pftp      | 207K                  | 604K  | 3077  | 207K                  | 480K  | 2578  | 207K              | 473K  | 2824  |
| snoopy    | 62179                 | 213K  | 6877  | 62179                 | 193K  | 5670  | 62179             | 192K  | 5546  |
| leader    | 38863                 | 158K  | 113   | 38863                 | 51565 | 112   | 38863             | 51565 | 113   |
| sort      | 374238                | 1.53M | 177   | 374238                | 413K  | 176   | 374238            | 413K  | 177   |

with the standard distribution of Spin. We observed in only one case (RW6) a difference between the number of states generated by EdgeLean and TNF.

In most of the experiments, both of our algorithms explore roughly the same number of edges, considerably fewer than regular depth-first search. With respect to the stack size (and thus memory consumption), our algorithms are up to a thousand times better than regular DFS. Also, on many examples TNF uses much less space than EdgeLean, but the converse is never the case.

## References

1. Basten, T., Bošnački, D., Geilen, M.: Cluster-based Partial Order Reduction. *Automated Software Engineering* 11(4), 365–402 (2004)
2. Bošnački, D., Elkind, E., Genest, B., Peled, D.: On Commutativity Based Edge Lean Search (full version), <http://perso.crans.org/~genest/BEGP.ps>
3. Păsăreanu, C.S., Dwyer, M.B., Huth, M.: Assume-Guarantee Model Checking of Software: A Comparative Case Study. In: Dams, D.R., Gerth, R., Leue, S., Massink, M. (eds.) *Theoretical and Practical Aspects of SPIN Model Checking*. LNCS, vol. 1680, Springer, Heidelberg (1999)
4. Clarke, E., Grumberg, O., Peled, D.: *Model Checking*. MIT Press, Cambridge (2000)
5. Godefroid, P.: *Partial-Order Methods for the Verification of Concurrent Systems – An Approach to the State-Explosion Problem*, PhD thesis, University of Liege, Computer Science Department (November 1994)
6. Godefroid, P., Wolper, P.: Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties. In: Larsen, K.G., Skou, A. (eds.) *CAV 1991*. LNCS, vol. 575, pp. 176–185. Springer, Heidelberg (1991)
7. Godefroid, P., Holzmann, G., Pirottin, D.: State-Space Caching Revisited. *Formal Methods in System Design* 7(3), 227–242 (1995)
8. Gouda, K., Zaki, M.J.: Efficiently Mining Maximal Frequent Itemsets. In: *IEEE International Conference on Data Mining (ICDM '01)*, pp. 163–170 (2001)
9. Holzmann, G.: *The SPIN Model Checking*. Addison Wesley, Reading (2003)

10. Koyuturk, M., Grama, A., Szpankowski, W.: An Efficient Algorithm for Detecting Frequent Subgraphs in Biological Networks. *Bioinformatics* 20, i200–i207 (2004)
11. Mazurkiewicz, A.: Trace semantics. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) *Advances in Petri Nets*. LNCS, vol. 255, pp. 279–324. Springer, Heidelberg (1986)
12. Ochmanski, E.: Languages and Automata. In: Diekert, V., Rozenberg, G. (eds.) *The Book of Traces*, pp. 167–204 (1995)
13. Peled, D.: Combining Partial Order Reductions with On-the-fly Model-Checking. In: Dill, D.L. (ed.) *CAV 1994*. LNCS, vol. 818, pp. 377–390. Springer, Heidelberg (1994)
14. Valmari, A.: A Stubborn Attack on State Explosion. *Formal Methods in System Design* 1(4), 297–322 (1992)

# Commitment Under Uncertainty: Two-Stage Stochastic Matching Problems

Irit Katriel\*, Claire Kenyon-Mathieu, and Eli Upfal\*\*

Brown University  
{irit,claire,eli}@cs.brown.edu

**Abstract.** We define and study two versions of the bipartite matching problem in the framework of two-stage stochastic optimization with recourse. In one version the uncertainty is in the second stage costs of the edges, in the other version the uncertainty is in the set of vertices that needs to be matched. We prove lower bounds, and analyze efficient strategies for both cases. These problems model real-life stochastic integral planning problems such as commodity trading, reservation systems and scheduling under uncertainty.

## 1 Introduction

Two-stage stochastic optimization with recourse is a popular model for hedging against uncertainty. Typically, part of the input to the problem is only known probabilistically in the first stage, when decisions have a low cost. In the second stage, the actual input is known but the costs of the decisions are higher. We then face a delicate tradeoff between speculating at a low cost vs. waiting for the uncertainty to be resolved.

This model has been studied extensively for problems that can be modeled by linear programming (sometimes using techniques such as Sample Average Approximation (SAA) when the linear program (LP) is too large.) Recently there has been a growing interest in 2-stage stochastic combinatorial optimization problems [1,2,6,12,19,20,21,22,24]. Since an LP relaxation does not guarantee an integer solution in general, one can either try to find an efficient rounding technique [11] or develop a purely combinatorial approach [5,8]. In order to develop successful algorithmic paradigms in this setting, there is an ongoing research program focusing on classical combinatorial optimization problems [23]: set cover, minimum spanning tree, Steiner tree, maximum weight matching, facility location, bin packing, multicommodity flow, minimum multicut, knapsack, and others. In this paper, we aim to enrich this research program by adding a basic combinatorial optimization problem to the list: the minimum cost maximum bipartite matching problem. The task is to buy edges of a bipartite graph which together contain a maximum-cardinality matching in the graph. We examine

---

\* Supported in part by NSF awards DMI-0600384 and ONR Award N000140610607.

\*\* Supported in part by NSF awards CCR-0121154 and DMI-0600384, and ONR Award N000140610607.

two variants of this problem. In the first, the uncertainty is in the second stage edge-costs, that is, the cost of an edge can either grow or shrink in the second stage. In the second variant, all edges become more expensive in the second stage, but the set of nodes that need to be matched is not known.

Here are some features of minimum cost maximum bipartite matching that make this problem particularly interesting. First, it is not subadditive: the union of two feasible solutions is not necessarily a solution for the union of the two instances. In contrast, most previous work focused on subadditive structures, with the notable exception of Gupta and Pál’s work on stochastic Steiner Tree [9]. Second, the solutions to two partial instances may interfere with one another in a way that seems to preclude the possibility of applying cost-sharing techniques associated with the scenario-sampling based algorithms [9,10]. This intuitively makes the problem resistant to routine attempts, and indeed, we confirm this intuition by proving a lower bound which is stronger than what is known<sup>1</sup> for the sub-additive problems: in Theorem 5, we prove a hardness of approximation result in the setting where the second-stage scenarios are generated by choosing vertices independently. It is therefore natural that our algorithms yield upper bounds which are either rather weak (Theorem 2, Part 1) or quite specialized (Theorem 7). To address this issue, we relax the constraint that the output be a maximum matching, and consider bicriteria results, where there is a tradeoff between the cost of the edges bought and the size of the resulting matching (Theorem 2, Part 2, and Theorem 8). This approach may be a way to circumvent hardness for other stochastic optimization problems as well.

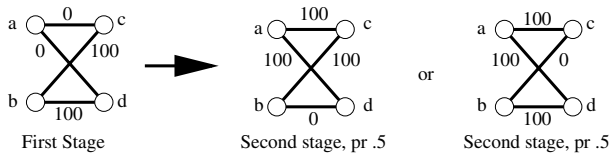
Although the primary focus of this work is stochastic optimization, another popular objective for the prudent investor is to minimize, not just the expected future cost, but the maximum future cost, over all possible future scenarios: that is the goal of robust optimization. We also prove a bicriteria result for robust optimization (Theorem 3). Guarding oneself against the worst case is more delicate than just working with expectations. The solution requires a different idea: preventing undesirable high-variance events by explicitly deciding, against the advice of the LP solution, to not buy expensive edges (To analyze this, the proof of Theorem 3 involves some careful rounding.) This general idea might be applicable to other problems as well.

We note that within two-stage stochastic optimization with recourse, matching has been studied before [15]. However, the problem studied here is very different: there, the goal was to construct a maximum weight matching instead of the competing objective of large size and small cost; moreover the set of edges bought by the algorithm had to form exactly a matching instead of just contain a matching. In Figure 1, we give an example illustrating the difference between requiring equality with a matching or containment of a matching.

Our main goal in this paper is to further fundamental understanding of the theory of stochastic optimization; however, we note that a conceivable

---

<sup>1</sup> To the best of our knowledge, all previous hardness results hold only when the second stage scenarios are given explicitly, i.e., when only certain combinations of parameter settings are possible.



**Fig. 1.** An example in which buying edges speculatively can help

application of this problem is commodity transactions, which can be viewed as a matching between supply and demand. When the commodity is indivisible, the set of possible transactions can be modeled as a weighted bipartite graph matching problem, where the weight of an edge represents the cost or profit of that transaction (including transportation cost when applicable). A trader tries to maximize profits or minimize total costs depending on her position in the transaction. A further tool that a commodity trader may employ to improve her income is timing the transaction. We model timing as a two-stage stochastic optimization problem with recourse: The trader can limit her risk by buying an option for a transaction at current information, or she can assume the risk and defer decisions to the second stage. Two common uncertainties in commodity transactions, price uncertainty and supply and demand uncertainty, correspond to the two stochastic two-stage matching problems mentioned above: finding minimum weight maximum matching with uncertain edge costs, and finding maximum matching with uncertain matching vertices. Similar decision scenarios involving matchings also show up in a variety of other applications such as scheduling and reservation systems.

Our results are summarized in the following table. We first prove (Theorem [1](#)) that, with explicit scenarios, the uncertain matching vertices case is in fact a special case of the uncertain edge costs case. Then, it suffices to prove upper bounds for the more general variant and lower bounds for the restricted one. For the problem of minimizing the expected cost of the solution, we show an approximability lower bound of  $\Omega(\log n)$ . We then describe an algorithm that finds a maximum matching in the graph at a cost which is an  $n^2$ -approximation for the optimum. We then show that by relaxing the demand that the algorithm constructs a maximum matching, we can “beat” the lower bound: At a cost of at most  $1/\beta$  times the optimum, we can match at least  $n(1 - \beta)$  vertices. Furthermore, we show that a similar bicriteria result holds also for the robust version of the problem, i.e., when we wish to minimize the worst-case cost.

With independent choices in the second-stage scenarios, our main contribution is the lower bound. The reduction of Theorem [1](#) does not apply, but we prove, for both types of uncertainty, that it is NP-hard to approximate the problem within better than a certain constant factor. We also prove an upper bound for a special case of the uncertain matching vertices variant.

| Input:                      | Explicit Scenarios  |  | Independent Choices  |
|-----------------------------|---|--|--|
| Criteria:                   | Expected Cost   | Worst-Case Cost  | Expected Cost  |
| Uncertain edge costs        | <ul style="list-style-type: none"> <li>• <math>n^2</math>-approximation of the cost to get a maximum matching [Theorem 2, part 1]</li> <li>• <math>1/\beta</math>-approximation of the cost to match at least <math>n(1 - \beta)</math> vertices [Theorem 2, part 2]</li> <li>• Same hardness results as below [Theorem 1]</li> </ul> | factor $1/\beta$ approximation of the cost to match at least $n(1 - \beta)$ vertices [Theorem 3] | NP-hard to approximate within a certain constant [Theorem 6]   |
| Uncertain matching vertices | <ul style="list-style-type: none"> <li>• <math>\Omega(\log n)</math> approximability lower bound [Theorem 4, Part 1]</li> <li>• NP-hard already for two scenarios [Theorem 4, Part 2]</li> <li>• Same upper bounds as above [Theorem 1]</li> </ul>  | As above [Theorem 1]   | <ul style="list-style-type: none"> <li>• As above [Theorem 5]</li> <li>• approximation for a special case [Theorem 7]</li> </ul> |

## 2 Explicit Scenarios

In this section, we assume that we have an explicit list of possible scenarios for the second stage.

*Uncertain edge costs.* Given a bipartite graph  $G = (A, B, E)$ , we can buy edge  $e$  in the first stage at cost  $C_e \geq 0$ , or we can buy it in the second stage at cost  $C_e^s \geq 0$  determined by the scenario  $s$ . The input has an explicit list of scenarios, and known edge costs ( $c_e^s$ ) in scenario  $s$ . For uncertain edge costs, without loss of generality we can assume that  $|A| = |B| = n$  and that  $G$  has a perfect matching (PM). Indeed, there is an easy reduction from the case where the maximum matching has size  $k$ : just create a new graph by adding a set  $A'$  of  $n - k$  vertices on the left side, a set  $B'$  of  $n - k$  vertices on the right side, and edges between all vertex pairs in  $A' \times B$  and in  $A \times B'$ , with cost 0.

In the *stochastic optimization* setting, the algorithm also has a known second stage distribution: scenario  $s$  occurs with probability  $\Pr(s)$ . The goal is, in time polynomial in both the size of the graph and the number of scenarios, to minimize the *expected* cost; if  $E_1$  denotes the set of edges bought in the first stage and  $E_2^s$  the set of edges bought in the second stage under scenario  $s$ , then:

$$\text{OPT}_1 = \min_{E_1, E_2^s} \left\{ \sum_{s \in S} \Pr(s) \left( \sum_{e \in E_1} C_e + \sum_{e \in E_2^s} C_e^s \right) : \forall s, E_1 \cup E_2^s \text{ has a PM} \right\} \quad (1)$$

Stochastic optimization with uncertain edge costs has been studied for many problems, see for example [10, 17].

In the *robust optimization* setting, the goal is to minimize the maximum cost (instead of the expected cost):

$$\text{OPT}_2 = \min_{E_1, E_2^s} \left\{ \max_{s \in S} \left( \sum_{e \in E_1} C_e + \sum_{e \in E_2^s} C_e^s \right) : \forall s, E_1 \cup E_2^s \text{ has a PM} \right\} \quad (2)$$

Robust optimization with uncertain edge costs has also been studied for many problems, see for example [4].

*Uncertain activated vertices.* In this variant of the problem, there is a known distribution over scenarios  $s$ , each being defined by a set  $B_s \subset B$  of *active* vertices that are allowed to be matched in that scenario. Each edge costs  $c_e$  today (before  $B_s$  is known) and  $\tau c_e$  tomorrow, where  $\tau > 1$  is the *inflation parameter*. As in Expression [1], the goal is to minimize the expected cost, i.e.,

$$\text{OPT}_3 = \{ C(E_1) + \tau \sum_{s \in S} \Pr(s) C(E_2^s) : \forall s, E_1 \cup E_2^s \text{ contains max matching of } (A, B_s, E \cap (A \times B_s)) \} \quad (3)$$

Stochastic optimization with uncertain activated vertices has also been previously studied for many problems, see for example [9]. There is a similar expression for robust optimization with uncertain activated vertices.

**Theorem 1 (Reduction).** *The two-stage stochastic matching problem with uncertain activated vertices and explicit second-stage scenarios ( $\text{OPT}_3$ ) reduces to the case of uncertain edge costs and explicit second-stage scenarios ( $\text{OPT}_1$ ).*

*Proof.* We give an approximation preserving reduction. Given an instance with stochastic matching vertices, we transform it to an instance of the problem with stochastic edge-costs, as follows. Assume that our input graph is  $G = (A, B, E)$  where  $A = \{a_1, \dots, a_{|A|}\}$  and  $B = \{b_1, \dots, b_{|B|}\}$ . We first add a set  $A' = a'_1, \dots, a'_{|B|}$  of  $|B|$  new vertices to  $A$ , and connect each  $a'_i$  with  $b_i$  by an edge. In other words, we generate the graph  $G' = (A \cup A', B, E \cup \{(a'_i, b_i) : 1 \leq i \leq |B|\})$ .

For the edges between  $A$  and  $B$ , edge costs are the same as in the original instance, in the first stage as well as the second stage. The costs on the edges between  $A'$  and  $B$  create the effect of selecting the activated vertices: For each  $(a'_i, b_i)$ , the first-stage cost is  $n^2W$ , and the second-stage cost is  $n^2W$  if  $b_i$  is active and 0 otherwise. Here,  $W$  is the maximum cost of an edge,  $nW$  is an upper bound on the cost of the optimal solution, and  $n^2W$  is large enough that any solution containing this edge cannot be an optimal, or even an  $n$ -approximate solution. Hence, a second-stage cost of 0 for  $(a'_i, b_i)$  allows  $b_i$  to be matched with  $a'_i$  for free, while a cost of  $nW$  forces  $b_i$  to be matched with a vertex from  $A$ . This concludes the reduction.  $\square$

From Theorem [1], it follows that our algorithms for uncertain edges costs (Theorems [2] and [3] below) imply corresponding algorithms for uncertain activated vertices, and that our lower bounds for uncertain activated vertices (Theorem [4] below) imply corresponding lower bounds for uncertain edge costs.

**Theorem 2 (Stochastic optimization upper bound)**

(1) *There is a polynomial-time deterministic algorithm for stochastic matching ( $\text{OPT}_1$ ) that constructs a perfect matching at expected cost is at most  $2n^2 \cdot \text{OPT}_1$ .*



(2) Given  $\beta \in (0, 1)$ , there is a polynomial-time randomized algorithm for stochastic matching ( $OPT_1$ ) that returns a matching whose cardinality, with probability  $1 - e^{-n}$  (over the random choices of the algorithm), is at least  $(1 - \beta)n$ , and whose overall expected cost is  $O(OPT_1/\beta)$ .

In particular, for any  $\epsilon > 0$  we get a matching of size  $(1 - \epsilon)n$  and cost  $O(OPT/\epsilon)$  in expectation. Note that by Theorem 4, we have to relax the constraint on the size anyway if we wish to obtain a better-than- $\log n$  approximation on the cost, so Part 2 of the Theorem is, in a sense, our best option.

*Proof.* The proof follows the general paradigm applied to stochastic optimization in recent papers such as [11]: formulate the problem as an integer linear program; solve the linear relaxation and use it to guide the algorithm; and use LP duality (König’s theorem, for our problem) for the analysis.

To define the integer program, let  $X_e$  indicate whether edge  $e$  is bought in the first stage, and for each scenario  $s$ , let  $Z_e^s$  (resp.  $Y_e^s$ ) indicate whether edge  $e$  is bought in the first stage (resp. in the second stage) and ends up in the perfect matching when scenario  $s$  materializes. We obtain:

$$\min \sum_{s \in S} \Pr(s) \left( \sum_e C_e X_e + \sum_e C_e^s Y_e^s \right) \text{ s.t. } \begin{cases} \sum_{e: v \in e} (Z_e^s + Y_e^s) = 1 \quad \forall v \in A \cup B, s \in S \\ Z_e^s \leq X_e \quad \forall e \in E, s \in S \\ X_e, Y_e^s, Z_e^s \in \{0, 1\} \quad \forall e \in E, s \in S. \end{cases}$$

The algorithm solves the standard LP relaxation, in which the last set of constraints is replaced by  $0 \leq X_e, Y_e^s, Z_e^s \leq 1$ . Let  $(X_e, Z_e^s, Y_e^s)$  denote the optimal solution of the LP. Now the proof of the two parts of the theorem diverges:

*Proof of part 1.* In the first stage, buy every edge  $e$  such that  $X_e \geq 1/(2n^2)$ . In the second stage, under scenario  $s$ , buy every edge  $e$  such that  $Y_e^s \geq 1/(2n^2)$ . Finally, output a maximum matching of the set of edges bought. The analysis, which relies on Hall’s theorem, is in [13].

*Proof of part 2.* In the first stage, buy each edge  $e$  with probability  $1 - e^{-X_e \alpha}$ . In the second stage under scenario  $s$ , buy each edge  $e$  with probability  $1 - e^{-Y_e^s \alpha}$ , where  $\alpha = 8 \ln(2)/\beta$ . Finally, output a maximum matching of the set of edges bought. The analysis, which relies on König’s theorem, is in [13].  $\square$

**Theorem 3 (Robust optimization).** *Given  $\beta \in (0, 1)$ , there is a polynomial-time randomized algorithm for robust matching ( $OPT_2$ ) with  $t$  scenarios that returns a matching s.t. with probability at least  $1 - 2/n$  (over the random choices of the algorithm), the following holds: In every scenario, the algorithm incurs cost  $O(OPT_2(1 + \ln(t)/\ln(n))/\beta)$  and outputs a matching of cardinality at least  $(1 - \beta)n$ .*

*Proof.* We detail this proof, which is the most interesting one in this section. The integer programming formulation is similar to the one used to prove Theorem 2. More specifically, let  $X_e$  indicate whether edge  $e$  is bought in the first stage, and for each scenario  $s$ , let  $Z_e^s$  (resp.  $Y_e^s$ ) indicate whether edge  $e$  is bought in the

first stage (resp. in the second stage) and ends up in the perfect matching when scenario  $s$  materializes. We obtain:

$$\min W \text{ s.t. } \begin{cases} \sum_{e:v \in e} (Z_e^s + Y_e^s) = 1 & \forall v \in A \cup B \text{ and } \forall s \in S \\ Z_e^s \leq X_e & \forall e \in E \text{ and } s \in S \\ \sum_e [C_e X_e + C_e^s Y_e^s] \leq W & \forall s \in S \\ X_e, Y_e^s, Z_e^s \in \{0, 1\} & \forall e \in E \text{ and } s \in S. \end{cases}$$

The algorithm solves the standard LP relaxation, in which the last set of constraints is replaced by  $0 \leq X_e, Y_e^s, Z_e^s \leq 1$ . Let  $w, (x_e), (y_e^s), (z_e^s)$  denote the optimal solution of the LP. Let  $\alpha = 8 \ln(2)/\beta$  again, and let  $T = 3 \ln n$ .

- In the first stage, relabel the edges so that  $c_1 \geq c_2 \geq \dots$ . Let  $t_1$  be maximum such that  $x_1 + x_2 + \dots + x_{t_1} \leq T$ . For every  $j > t_1$ , buy edge  $j$  with probability  $1 - e^{-x_j \alpha}$ . (Do not buy any edge  $j \leq t_1$ .)
- In the second stage, relabel the remaining edges so that  $c_1^s \geq c_2^s \geq \dots$ . Let  $t_2$  be maximum such that  $y_1^s + y_2^s + \dots + y_{t_2}^s \leq T$ . For every  $j > t_2$ , buy edge  $j$  with probability  $1 - e^{-y_j^s \alpha}$ . (Do not buy any edge  $j \leq t_2$ .)

Finally, the algorithm computes and returns a maximum matching of the set of edges bought.

We note that this construction and the rounding used in the analysis are almost identical to the construction used in strip-packing [14]. The analysis of the cost of the edges bought is the difficult part. We first do a slight change of notations. The cost can be expressed as the sum of at most  $2m$  random variables (at most  $m$  in each stage). Let  $a_1 \geq a_2 \geq \dots$  be the multiset  $\{c_e\} \cup \{c_e^s\}$ , along with the corresponding probabilities  $p_i$  ( $p_i = 1 - e^{-x_e \alpha}$  if  $a_i = c_e$  is a first-stage cost, and  $p_i = 1 - e^{-y_e^s \alpha}$  if  $a_i = c_e^s$  is a second-stage cost.) Let  $X_i$  be the binary variable with expectation  $p_i$ . Clearly, the cost incurred by the algorithm can be bounded above by  $X = \sum_{i > t^*} a_i X_i$ , where  $t^*$  is maximum such that  $p_1 + \dots + p_{t^*} \leq T$ .

To prove a high-probability bound on  $X$ , we will partition  $[1, 2m]$  into intervals to define groups. The first group is just  $[1, t^*]$ , and the subsequent groups are defined in greedy fashion, with group  $[j, \ell]$  defined by choosing  $\ell$  maximum so that  $\sum_{i \in [j, \ell]} p_i \leq T$ . Let  $G_1, G_2, \dots, G_r$  be the resulting groups. We have:

$$X \leq \sum_{\ell \geq 2} \sum_{i \in G_\ell} a_i X_i \leq \sum_{\ell \geq 2} \sum_{i \in G_\ell} (\max_{G_\ell} a_i) X_i \leq \sum_{\ell \geq 2} \sum_{i \in G_\ell} (\min_{G_{\ell-1}} a_i) X_i \leq \sum_{\ell \geq 1} (\min_{G_\ell} a_i) \sum_{i \in G_{\ell+1}} X_i.$$

On the other hand, (using the inequality  $1 - e^{-Z} \leq Z$ ), the optimal value  $\text{OPT}^*$  of the LP relaxation satisfies:

$$\alpha \text{OPT}^* \geq \sum_i a_i p_i \geq \sum_{\ell \geq 1} \sum_{i \in G_\ell} (\min_{G_\ell} a_i) p_i \geq \sum_{\ell \geq 1} (\min_{G_\ell} a_i) (T - 1).$$

It remains, for each group  $G_\ell$ , to apply a standard Chernoff bound to bound the sum of the  $X_i$ 's in  $G_\ell$ , and use union bounds to put these results together and yield the statement of the theorem [13]. □

We note that the proof of Theorem 3 can also be extended to the setting of Theorem 2 to prove a high probability result: For scenario  $s$ , with probability at least  $1 - 2/n$  over the random choices of the algorithm, the algorithm incurs cost  $O(\text{OPT}_s/\beta)$  and outputs a matching of cardinality at least  $(1 - \beta)n$ , where  $\text{OPT}_s = \sum_{E_1} C_e + \sum_{E_2^s} C_e^s$ .

Finally, we can show two hardness of approximation results for the explicit scenario case.

#### Theorem 4 (Stochastic optimization lower bound)

1. *There exists a constant  $c > 0$  such that Expression  $\text{OPT}_3$  (Eq 3) is NP-hard to approximate within a factor of  $c \ln n$ .*
2. *Expression  $\text{OPT}_3$  (Eq 3) is NP-hard to compute, even when there are only two scenarios and  $\tau$  is bounded.*

*Proof.* To prove Part 1, we show that when  $\tau \geq n^2$ , Expression (3) is at least as hard to approximate as Minimum Set Cover: Given a universe  $S = \{s_1, \dots, s_n\}$  of elements and a collection  $C = \{c_1, \dots, c_k\}$  of subsets of  $S$ , find a minimum-cardinality subset  $\mathcal{SC}$  of  $C$  such that for every  $1 \leq i \leq n$ ,  $s_i \in c_j$  for some  $c_j \in \mathcal{SC}$ . It is known that there exists a constant  $c > 0$  such that approximating Minimum Set-Cover to within a factor of  $c \ln n$  is NP-hard [18].

Given an instance  $(S = \{s_1, \dots, s_n\}; C = \{c_1, \dots, c_k\})$  of Minimum Set-Cover, we construct an instance of the two-stage matching problem with stochastic matching vertices as follows. The graph contains  $|S| + 3|C|$  vertices: for every element  $s_i \in S$  there is a vertex  $u_i$ ; for every set  $c_j \in C$ , there are three vertices  $x_j, y_j$ , and  $z_j$  connected by a path  $(x_j, y_j), (y_j, z_j)$ . For every set  $c_j$  and element  $s_i$  which belongs to  $c_j$ , we have the edge  $(z_j, u_i)$ . It is easy to see that the graph is bipartite. The first-stage edge costs are 1 for an  $(x_i, y_i)$  edge costs and 0 for the other edges. The second-stage costs are equal to the first-stage costs, multiplied by  $\tau$ . There are  $n$  equally likely second-stage scenarios: In scenario  $i$  the vertices in  $\{y_1, \dots, y_k\} \cup \{u_i\}$  are active. In [13] we show that the optimal solution to the stochastic matchings instance buys, in the first stage, the edge  $(x_j, y_j)$  for each set  $c_j$  in some minimum set cover of the input.

The proof of Part 2 is by reduction from the Simultaneous Matchings [7] problem and is also in [13].  $\square$

### 3 Implicit Scenarios

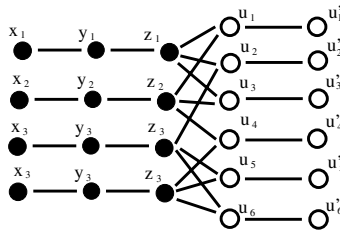
Instead of having an explicit list of scenarios for the second stage, it is common to have instead an implicit description: in the case of uncertain activated vertices, a natural stochastic model is the one in which each vertex is active in the second stage with some probability  $p$ , independently of the status of the other nodes. Due to independence, we get that although the total number of possible scenarios can be exponentially large, there is a succinct description consisting of simply specifying the activation probability of each node. In this case, we can no longer be certain that the second-stage graph contains a perfect matching even if the input graph does, so the requirement is, as stated above, to find the largest possible matching. We first prove an interesting lower bound.

### 3.1 Lower Bounds

**Theorem 5.** *Stochastic optimization with uncertain vertex set is NP-hard to approximate within a certain constant, even with independent vertex activation.*

*Proof.* We detail this proof, which is the most interesting of our lower bounds. We will use a reduction from Minimum 3-Set-Cover(2), the special case of Minimum Set-Cover where each set has cardinality 3 and each element belongs to two sets [16]. This variant is NP-hard to approximate to within a factor of 100/99 [3]. We will prove that approximating Expression (3) to within a factor of  $\beta$  is at least as hard as approximating 3-set-cover(2) to within a factor of  $\gamma = \beta(1 + (3p^2(1 - p) + 2p^3)\tau)$ . The theorem follows by setting  $p$  to be a constant in the interval  $[0, 0.0033]$  and  $\tau = 1/p$ , because then  $3p(1 - p) + 2p^2 < 1/99$ .

Given an instance  $(S = \{s_1, \dots, s_n\}; C = \{c_1, \dots, c_k\})$  of 3-set-cover(2), we construct an instance of the two-stage matching problem with uncertain activated vertices as follows (see Figure 2). The graph contains  $2|S| + 3|C|$  vertices: for every element  $s_i \in S$  there are two vertices  $u_i, u'_i$  connected by an edge whose first stage cost is 1; for every set  $c_j \in C$ , there are three vertices  $x_j, y_j,$  and  $z_j$  connected by a path  $(x_j, y_j), (y_j, z_j)$ . For every set  $c_j$  and element  $s_i$  which belongs to  $c_j$ , we have the edge  $(z_j, u_i)$ . It is easy to see that the graph is bipartite. The first-stage edge costs are 1 for an  $(x_i, y_i)$  edge and 0 for the other edges. The second-stage costs are equal to the first-stage costs, multiplied by  $\tau$ . In the second-stage scenarios, each vertex  $u_i$  is active with probability  $p$  and each  $y_i$  is active with probability 1.



**Fig. 2.** The graph obtained from the 3-Set-Cover(2) instance  $\{s_1, s_2, s_3\}, \{s_1, s_3, s_4\}, \{s_2, s_5, s_6\}, \{s_4, s_5, s_6\}$

If  $p > 1/\tau$ , then buying all  $(u_i, u'_i)$  edges in the first stage at cost  $n$  is optimal. To see why, assume that an algorithm spends  $n' < n$  in the first stage. In the second stage, the expected number of active vertices that cannot be matched is at least  $(n - n')p$  and the expected cost of matching them is  $\tau(n - n')p > (n - n')$ . We assume in the following that  $p \leq 1/\tau$ .

Consider a minimum set cover  $\mathcal{SC}$  of the input instance. Assume that in the first stage we buy (at cost 1) the edge  $(x_j, y_j)$  for every set  $c_j \in \mathcal{SC}$ . In the second stage, let  $I$  be the set of active vertices and find, in a way to be described shortly, a matching  $M_I$  between a subset  $I'$  of  $I$  and the vertex-set  $\{z_j : c_j \in \mathcal{SC}\}$ ,

using  $(z_j, u_i)$ -edges from the graph. Buy the edges in  $M_I$  (at cost 0). For every  $i \in I \setminus I'$ , buy the edge  $(u_i, u'_i)$  at cost  $\tau$ . Now, all active  $u_i$  vertices are matched, and it remains to ensure that the  $y$ -vertices are matched as well. Assume that  $y_j$  is unmatched. If  $z_j$  is matched with some  $u_i$  node, this is because  $c_j \in \mathcal{SC}$ , so we bought the edge  $(x_j, y_j)$  in the first stage and can now use it at no additional cost. Otherwise, we buy the edge  $(y_j, z_j)$  at cost 0. The second stage has cost equal to  $\tau$  times the cardinality of  $I \setminus I'$  and the first stage has cost equal to the cardinality of the set cover. The matching  $M_I$  is found in a straightforward manner: Given  $\mathcal{SC}$ , each element chooses exactly one set among the sets covering it, and, if it turns out to be active, will only try to be matched to that set. Each set in the set cover will be matched with one element, chosen arbitrarily among the active vertices who try to be matched with it.

To calculate the expected cost of matching the vertices of  $I - I'$ , consider a set in  $\mathcal{SC}$ . It has 3 elements, and is chosen by at most 3 of them. Assume that it is chosen by all 3. With probability  $(1 - p)^3 + 3p(1 - p)^2$ , at most one of them is active and no cost is incurred in the second stage. With probability  $3p^2(1 - p)$ , two of them are active and a cost of  $\tau$  is incurred. With probability  $p^3$ , all three of them are active and a cost of  $2\tau$  is incurred, for an expected cost of  $(3p^2(1 - p) + 2p^3)\tau$ . If the set is chosen by two elements, the expected cost is at most  $p^2\tau$ , and if it is chosen by fewer, the expected cost is 0. Thus in all cases the expected cost of matching  $I \setminus I'$  is bounded by  $|\mathcal{SC}|(3p^2(1 - p) + 2p^3)\tau$ . With a cost of  $|\mathcal{SC}|$  for the first stage, we get that the total cost of the solution is at most  $|\mathcal{SC}|(1 + (3p^2(1 - p) + 2p^3)\tau)$ .

On the other hand, let  $\mathcal{M}_1$  be the set of cost-1 edges bought in the first stage. Let an  $(x_i, y_i)$  edge represent the set  $c_i$  and let a  $(u_i, u'_i)$  edge represent the singleton set  $\{s_i\}$ . Now, assume that  $\mathcal{M}_1$  does not correspond to a set cover of the input instance. Let  $x$  be the number of elements which are not covered by the sets corresponding to  $\mathcal{M}_1$  and let  $X$  be the number of active elements among those  $x$ . In the second stage, the algorithm will have to match each uncovered element vertex  $u_i$ , either by its  $(u_i, u'_i)$  edge (at cost  $n$ ) or by a  $(z_j, u_i)$  edge for some set  $c_j$  where  $s_i \in c_j$ . In the latter case, it would have to buy the edge  $(x_i, y_i)$ , again at cost  $n$ . The second stage cost, therefore, is at least  $Xn$ . But the expected value of  $X$  is  $x/n$ , thus the total expected cost is at least  $|\mathcal{M}_1| + x$ . Since we could complete  $\mathcal{M}_1$  into a set cover by adding at most one set per uncovered element, we have  $x + |\mathcal{M}_1| \geq |\mathcal{SC}|$ .

In summary, we get that Expression (3) satisfies

$$|\mathcal{SC}| \leq \text{OPT} \leq |\mathcal{SC}|(1 + (3p^2(1 - p) + 2p^3)\tau).$$

This means that if we can approximate our problem within a factor of  $\beta$ , then we can approximate Minimum 3-Set-Cover(2) within a factor of  $\gamma = \beta(1 + (3p^2(1 - p) + 2p^3)\tau)$ , and the theorem follows.  $\square$

Using similar ideas, we prove the following related result in [13].

**Theorem 6.** *The case of uncertain, independent, edge costs is NP-hard to approximate within a certain constant.*

### 3.2 Upper Bound in a Special Case

We can show that when  $c_e = 1$  for all  $e \in E$ , it is possible to construct a perfect matching cheaply when the graph has certain properties. We study the case in which  $B$  is significantly larger than  $A$ .

**Theorem 7.** *Assume that the graph contains  $n$  vertex-disjoint stars  $s_1, \dots, s_n$  such that star  $s_i$  contains  $d = \max\{1, \ln(\tau p)\} / \ln(1/(1-p)) + 1$  vertices from  $B$  and is centered at some vertex of  $A$ . Then there is an algorithm whose running time is polynomial in  $n$  and which returns a maximum-cardinality matching of the second stage graph, whose expected cost is  $O(OPT_3 \cdot \min\{1, \ln(\tau p)\})$ .*

To prove this, let  $A = \{a_1, \dots, a_n\}$  and  $B = \{b_1, \dots, b_m\}$ . Let  $E_1$  be the edges in the stars. Let  $B_2$  be the vertices which are active in the second stage. Here is the algorithm. In the first stage, if  $\tau p \leq \epsilon$  then the algorithm buys nothing; else, the algorithm buys all edges of  $E_1$ , paying  $nd$ . In the second stage, the algorithm completes its set of edges into a perfect matching in the cheapest way possible. It remains to show that the expected cost of the second stage is low, compared to the optimal cost. We do this by showing that the number of edges bought in the second stage is proportional to the number of nodes of  $A$  that have at most one active node in their stars, and that there are few such nodes. The details are in [13].

### 3.3 Generalization: The Black Box Model

With independently activated vertices, the number of scenarios is extremely large, and so solving an LP of the kind described in previous sections is prohibitively time-consuming. However, in such a situation there is often a *black box* sampling procedure that provides, in polynomial time, an unbiased sample of scenarios; then one can use the SAA method to simulate the explicit scenarios case, and, if the edge cost distributions have bounded second moment, one can extend the analysis so as to obtain a similar approximation guarantee. The main observation is that the value of the LP defined by taking a polynomial number of samples of scenarios tightly approximates the value of the LP defined by taking all possible scenarios. An analysis similar to [5] gives:

**Theorem 8.** *Consider a two-stage edge stochastic matching problem with (1) a polynomial time unbiased sampling procedure and (2) edge cost distributions have bounded second moment. For any constants  $\epsilon > 0$  and  $\delta, \beta \in (0, 1)$ , there is a polynomial-time randomized algorithm that outputs a matching whose cardinality is at least  $(1-\beta)n$  and, with probability at least  $1-\delta$  (over the choices of the black box and of the algorithm), incurs expected cost  $O(OPT/\beta)$  (where the expectation is over the space of scenarios).*

## References

1. Birge, J., Louveaux, F.: Introduction to Stochastic Programming. Springer, Heidelberg (1997)
2. Charikar, M., Chekuri, C., Pal, M.: Sampling bounds for stochastic optimization. In: APPROX-RANDOM, pp. 257–269 (2005)

3. Chlebík, M., Chlebíková, J.: Inapproximability results for bounded variants of optimization problems. In: Lingas, A., Nilsson, B.J. (eds.) FCT 2003. LNCS, vol. 2751, pp. 27–38. Springer, Heidelberg (2003)
4. Dhamdhere, K., Goyal, V., Ravi, R., Singh, M.: How to pay, come what may: Approximation algorithms for demand-robust covering problems. In: FOCS, pp. 367–378 (2005)
5. Dhamdhere, K., Ravi, R., Singh, M.: On two-stage stochastic minimum spanning trees. In: Jünger, M., Kaibel, V. (eds.) Integer Programming and Combinatorial Optimization. LNCS, vol. 3509, pp. 321–334. Springer, Heidelberg (2005)
6. Dye, S., Stougie, L., Tomasgard, A.: The stochastic single resource service-provision problem. *Naval Research Logistics* 50, 257–269 (2003)
7. Elbassioni, K.M., Katriel, I., Kutz, M., Mahajan, M.: Simultaneous matchings. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 106–115. Springer, Heidelberg (2005)
8. Flaxman, A.D., Frieze, A.M., Krivelevich, M.: On the random 2-stage minimum spanning tree. In: SODA, pp. 919–926 (2005)
9. Gupta, A., Pál, M.: Stochastic steiner trees without a root. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1051–1063. Springer, Heidelberg (2005)
10. Gupta, A., Pál, M., Ravi, R., Sinha, A.: Boosted sampling: approximation algorithms for stochastic optimization. In: STOC, pp. 417–426. ACM, New York (2004)
11. Gupta, A., Ravi, R., Sinha, A.: An edge in time saves nine: LP rounding approximation algorithms for stochastic network design. In: FOCS, pp. 218–227 (2004)
12. Immorlica, N., Karger, D., Minkoff, M., Mirrokni, V.S.: On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In: SODA, pp. 691–700 (2004)
13. Katriel, I., Kenyon-Mathieu, C., Upfal, E.: Commitment under uncertainty: Two-stage stochastic matching problems. *ECCC* (2007). <http://eccc.hpi-web.de/eccc/>
14. Kenyon, C., Rémla, E.: A near-optimal solution to a two-dimensional cutting stock problem. *Math. Oper. Res.* 25(4), 645–656 (2000)
15. Kong, N., Schaefer, A.J.: A factor 1/2 approximation algorithm for two-stage stochastic matching problems. *Eur. J. of Operational Research* 172, 740–746 (2006)
16. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation, and complexity classes. *J. of Computing and System Sciences* 43, 425–440 (1991)
17. Ravi, R., Sinha, A.: Hedging uncertainty: Approximation algorithms for stochastic optimization problems. In: Bienstock, D., Nemhauser, G.L. (eds.) *Integer Programming and Combinatorial Optimization*. LNCS, vol. 3064, pp. 101–115. Springer, Heidelberg (2004)
18. Raz, R., Safra, S.: A sub-constant error-prob. low-degree test, and a sub-constant error-prob. PCP characterization of NP. In: STOC, pp. 475–484 (1997)
19. Shmoys, D.B., Sozio, M.: Approximation algorithms for 2-stage stochastic scheduling problems. In: IPCO (2007)
20. Shmoys, D.B., Swamy, C.: The sample average approximation method for 2-stage stochastic optimization (2004)
21. Shmoys, D.B., Swamy, C.: Stochastic optimization is almost as easy as deterministic optimization. In: FOCS, pp. 228–237 (2004)
22. Swamy, C., Shmoys, D.B.: The sampling-based approximation algorithms for multi-stage stochastic optimization. In: FOCS, pp. 357–366 (2005)
23. Swamy, C., Shmoys, D.B.: Algorithms column: Approximation algorithms for 2-stage stochastic optimization problems. *ACM SIGACT News* 37(1), 33–46 (2006)
24. Verweij, B., Ahmed, S., Kleywegt, A.J., Nemhauser, G., Shapiro, A.: The sample average approximation method applied to stochastic routing problems: a computational study. *Comp. Optimization and Applications* 24, 289–333 (2003)

# On the Complexity of Hard-Core Set Constructions

Chi-Jen Lu<sup>1</sup>, Shi-Chun Tsai<sup>2</sup>, and Hsin-Lung Wu<sup>2</sup>

<sup>1</sup> Institute of Information Science, Academia Sinica, Taipei, Taiwan  
cjlu@iis.sinica.edu.tw

<sup>2</sup> Department of Computer Science, National Chiao-Tung University, Hsinchu, Taiwan  
{sctsai,hsinlung}@csie.nctu.edu.tw

**Abstract.** We study a fundamental result of Impagliazzo (*FOCS'95*) known as the hard-core set lemma. Consider any function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  which is “mildly-hard”, in the sense that any circuit of size  $s$  must disagree with  $f$  on at least  $\delta$  fraction of inputs. Then the hard-core set lemma says that  $f$  must have a hard-core set  $H$  of density  $\delta$  on which it is “extremely hard”, in the sense that any circuit of size  $s' = O(s/(\frac{1}{\varepsilon^2} \log(\frac{1}{\varepsilon\delta})))$  must disagree with  $f$  on at least  $(1-\varepsilon)/2$  fraction of inputs from  $H$ .

There are three issues of the lemma which we would like to address: the loss of circuit size, the need of non-uniformity, and its inapplicability to a low-level complexity class. We introduce two models of hard-core set constructions, a strongly black-box one and a weakly black-box one, and show that those issues are unavoidable in such models.

First, we show that in any strongly black-box construction, one can only prove the hardness of a hard-core set for smaller circuits of size at most  $s' = O(s/(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}))$ . Next, we show that any weakly black-box construction must be inherently non-uniform — to have a hard-core set for a class  $G$  of functions, we need to start from the assumption that  $f$  is hard against a non-uniform complexity class with  $\Omega(\frac{1}{\varepsilon} \log |G|)$  bits of advice. Finally, we show that weakly black-box constructions in general cannot be realized in a low-level complexity class such as  $\text{AC}^0[p]$  — the assumption that  $f$  is hard for  $\text{AC}^0[p]$  is not sufficient to guarantee the existence of a hard-core set.

## 1 Introduction

One fundamental notion in complexity theory is the hardness of a function. Informally speaking, a function  $f$  is hard if any circuit of small size must fail to compute it correctly on some inputs. More precisely, we can characterize the hardness by parameters  $\delta$  and  $s$ , and say that  $f$  is  $\delta$ -hard (or has hardness  $\delta$ ) for size  $s$  if any circuit of size  $s$  must fail to compute  $f$  correctly on at least  $\delta$  fraction of inputs. An important question is: given a  $\delta$ -hard function for size  $s$ , can we transform it into a harder function with hardness  $\delta' > \delta$  for size about  $s$ ? This is known as the task of hardness amplification, and it plays a crucial role in the study of derandomization, in which one would like to obtain an extremely hard



function, with hardness  $(1 - \varepsilon)/2$ , so that it looks like a random function and can be used to construct pseudo-random generators [19,21,7,8,12]. One may wonder if the hardness of a function basically comes from a subset of density about  $\delta$ . So the question is: given any  $\delta$ -hard function for size  $s$ , is there always a subset of inputs of density about  $\delta$  on which  $f$  is extremely hard for circuits of size about  $s'$ ? A seminal result of Impagliazzo [7] answers this affirmatively. He showed that any  $\delta$ -hard function for size  $s$  indeed has a subset  $H$  of the inputs with density  $\delta$  on which  $f$  has hardness  $(1 - \varepsilon)/2$  for circuits of size  $s' = O(s/(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}))$ . Such a set  $H$  is called an  $\varepsilon$ -hard-core set for size  $s'$ .

In addition to answering a very basic question in complexity theory, the hard-core set lemma has found applications in learning theory [9] and cryptography [6], and has become an important tool in the study of pseudo-randomness. It can be used to provide an alternative proof of Yao's celebrated XOR Lemma [7], or to construct a pseudo-random generator directly from a mildly-hard function, bypassing the XOR lemma [12]. Recently, it has become a key ingredient in the study of hardness amplification for functions in NP [11,16,17,5]. In spite of its importance, there are some issues of the hard-core set lemma which are still not well understood and have become the bottlenecks in some applications. This calls for a more thorough study of the lemma.

The first issue is the loss of circuit size. Note that in Impagliazzo's result, the hardness on the hard-core set, although increased, is actually measured against circuits of a smaller size  $s'$ , as opposed to the initial size  $s$ . This loss of circuit size was later reduced by Klivans and Servedio [9] who showed the existence an  $\varepsilon$ -hard-core set of density  $\delta/2$  for size  $s' = O(s/(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}))$ . Then a natural question is: can the size  $s'$  be further improved to, say,  $\Omega(s)$ ? The second issue is non-uniformity. Note that even when one only wants to have a hard-core set which is hard for uniform algorithms, one still needs to start from a function which is hard for non-uniform circuits, or algorithms supplied with advices. In fact, this becomes the bottleneck in Trevisan's work of uniform hardness amplification for functions in NP [16,17], in which he showed how to amplify hardness from  $\delta = 1 - 1/\text{poly}(n)$  to  $\delta' = (1 - \varepsilon)/2$  against BPP algorithms, with  $\varepsilon = 1/\log^c n$  for a small constant  $c < 1$ . Note that ideally one would like to have the hardness amplified to  $(1 - \varepsilon)/2$  with  $\varepsilon \leq 1/\text{poly}(n)$ , and what prevents Trevisan from reaching this goal (or even with  $\varepsilon = 1/\log n$ ) is the large number (proportional to  $1/\varepsilon^2$ ) of advice bits needed by the hard-core set lemma. On the other hand, it is known that hardness amplification for functions in a higher complexity class, such as EXP, only requires  $O(\log(1/\varepsilon))$  bits of advice [12]. So a natural question is: can the number of advice bits needed in the hard-core set lemma be reduced? The third issue is that the lemma currently does not apply to a low-level complexity class such as  $\text{AC}^0[p]$ . That is, one needs to start from the assumption that  $f$  is hard for a complexity class which is high enough to include the majority function. Thus, an interesting question is: for any function  $f$  which is  $\delta$ -hard for  $\text{AC}^0[p]$ , does it always have an  $\varepsilon$ -hard-core set for  $\text{AC}^0[p]$ ?

All these three issues seem inherent in Impagliazzo's proof and they look difficult to resolve. One may wonder that perhaps they are indeed impossible to

avoid. However, proving such negative results appears to require proving circuit lower bounds, which seems to be far beyond our reach. Therefore, we would like to identify general and reasonable models for the hard-core set lemma in which such negative results can actually be proven.

*Black-Box Hard-Core Set Constructions.* The hard-core set lemma, when stated in the contrapositive way, basically says that given any function  $f$  with no hard-core set for small circuits (on any such subset  $H$ , there is a small circuit  $C_H$  with a good correlation with  $f$ ), one can find a small circuit  $C$  which is close to  $f$ . A closer look at Impagliazzo’s proof shows that the circuit  $C$  is simply the weighted majority on a small subset of those circuits  $C_H$ ’s. In fact, one can replace the class of small circuits  $C_H$ ’s by any class  $G$  of functions, and Impagliazzo’s proof shows that given any  $f$  with no hard-core set for functions in  $G$ , one can construct a function  $C$  close to  $f$  by taking a weighted majority on a small subset of functions in  $G$ . We call this type of argument a hard-core set construction, and note that  $C$  only uses those functions in  $G$  as an oracle.

This motivates us to define our first model of hard-core set constructions as follows. We say that a (non-uniform) oracle algorithm  $\text{DEC}^{(\cdot)}$  with a decision function  $D : \{0, 1\}^q \rightarrow \{0, 1\}$  realizes a *strongly black-box*  $(\delta, \varepsilon, k)$ -construction (of hard-core set) if the following holds. First,  $\text{DEC}$  will be given a family  $G = \{g_1, \dots, g_k\}$  of functions as oracle together with a multi-set  $I = \{i_1, \dots, i_q\}$  as advice, and for any input  $x$ , it will query the functions  $g_{i_1}, \dots, g_{i_q}$ , all at  $x$ , and then output  $D(g_{i_1}(x), \dots, g_{i_q}(x))$ . Moreover, it satisfies the property that for any  $G$  and for any  $f$  which has no  $\varepsilon$ -hard-core set of density  $\Omega(\delta)$  for  $G$ , there exists a multi-set  $I$  of size  $q$  such that the function  $\text{DEC}^{G,I}$  is  $\delta$ -close to  $f$  ( $\text{DEC}^{G,I}(x) \neq f(x)$  for at most  $\delta$  fraction of  $x$ ). We call  $q$  the query complexity of  $\text{DEC}$ , and observe that it relates to the loss of circuit size in the hard-core set lemma, with  $s' = O(s/q)$ . Note that the known hard-core set constructions [7,9] are in fact done in such a strongly black-box way.

Our second model of hard-core set constructions generalizes the first one by removing the constraint on how the algorithm  $\text{DEC}$  works; the algorithm  $\text{DEC}$  and its advice now are allowed to be of arbitrary form. We say that a (non-uniform) oracle algorithm  $\text{DEC}^{(\cdot)}$  realizes a *weakly black-box*  $(\delta, \varepsilon, k)$ -construction (of hard-core set) if the following holds. For any family  $G$  of  $k$  functions and for any function  $f$  which has no  $\varepsilon$ -hard-core set of density  $\Omega(\delta)$  for  $G$ , there exists an advice string  $\alpha$  such that  $\text{DEC}^{G,\alpha}$  is  $\delta$ -close to  $f$ .

*Our Results.* We have three results, which give negative answers to the three questions we raised before, with respect to our models of black-box constructions. Note that our lower bound for our second model (weakly black-box one) also holds for our first model as the first model is a special case of the second one.

Our first result shows that any *strongly* black-box  $(\delta, \varepsilon, k)$ -construction must require a query complexity of  $q = \Omega(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ . Our lower bound explains why it is very difficult to have a smaller loss of circuit size in the hard-core set lemma; in fact, any strongly black-box construction must suffer a loss of such a large

factor  $q$ . Note that our lower bound is tight as it is matched (up to a constant factor) by the upper bound from the construction of Klivans and Servedio [9].

Our second result shows that any *weakly* black-box  $(\delta, \varepsilon, k)$ -construction must require an advice of length  $\Omega(\frac{1}{\varepsilon} \log k)$ . This explains why it is difficult to have a uniform version of the hard-core set lemma; in fact, any weakly black-box construction is inherently non-uniform. Moreover, one cannot hope to improve Trevisan's uniform hardness amplification results [16,17] by reducing the number of advice bits needed in the hard-core set construction, unless one can come up with a non-black-box approach. Note that from the query upper bound of [9], one can immediately have an advice upper bound of  $O(\frac{1}{\varepsilon^2} (\log \frac{1}{\delta}) \log k)$ , which has a gap from our lower bound. It is not clear which bound can be further improved, but our feeling is that this upper bound may likely be improved.

Our third result shows that no *weakly* black-box  $(\delta, \varepsilon, k)$ -construction can be implemented in a low-level complexity class such as  $\text{AC}^0[p]$  for a prime  $p$ , when  $\delta < 1/20$  and  $\varepsilon \leq 1/n$ . More precisely, we show that the function DEC realizing such a black-box construction can be used to approximate the majority function, but on the other hand, the majority function cannot be approximated by an  $\text{AC}^0[p]$  circuit. Therefore one cannot have a hard-core set lemma for  $\text{AC}^0[p]$ , unless one can prove it in a non-black-box way.

*Bounds from Hardness Amplification.* There is no previous result directly on the lower bounds of hard-core set constructions. However, one can obtain such bounds from lower bounds for the task of hardness amplification [18,10]. This is because the hard-core set lemma can be used for hardness amplification, as shown in [7], and a closer inspection shows that a black-box hard-core set construction in fact yields a hardness amplification in a similar black-box model.

In particular, one can have the following. First, using a recent result of Viola [18], we can derive a lower bound of  $\min(\frac{1}{10\varepsilon}, \frac{n}{5 \log n})$  on the query complexity of any strongly black-box  $(\delta, \varepsilon, k)$ -construction. Note that this bound is always smaller than our bound. Second, we can use the result in [10] to derive an advice lower bound of  $\Omega(\log \frac{(1-\delta)^2}{\varepsilon})$  for any weakly black-box  $(\delta, \varepsilon, k)$ -construction. Note that this bound is exponentially worse than ours. Finally, we can use another result of Viola [18] to show that for any weakly black-box  $(\delta, \varepsilon, k)$ -construction, if the function DEC satisfies the additional condition that it only needs a short (logarithmic in the circuit size of DEC) advice, then it cannot be implemented by an  $\text{AC}^0[p]$  circuit. Note that this additional condition is not required in our result and our proof is much simpler. (On the other hand, under this additional condition, Viola achieved something stronger: such DEC can be used as oracle gates by an  $\text{AC}^0$  circuit to compute the majority function exactly.)

## 2 Preliminaries

For any  $n \in \mathbb{N}$ , let  $[n]$  denote the set  $\{1, \dots, n\}$  and let  $\mathcal{U}_n$  denote the uniform distribution over  $\{0, 1\}^n$ . For a finite set  $X$ , we will also use  $X$  to denote the uniform distribution over it when there is no confusion. For a string  $x \in \{0, 1\}^n$ ,

we let  $x_i$  denote the  $i$ -th bit of  $x$ . Let  $F_n$  denote the set of all Boolean functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . Let  $\text{SIZE}(s)$  be the class of Boolean functions computed by circuits of size  $s$ . Let  $\text{AC}^0[p](s)$  denote the class of Boolean functions computed by constant-depth circuits of size  $s$  equipped with  $\text{mod}_p$  gates (which output 0 exactly when the input bits sum to 0 modulo  $p$ ), and let  $\text{AC}^0[p] = \text{AC}^0[p](\text{poly}(n))$ . Given a multi-set (or simply a set)  $S$ , we let  $|S|$  denote the number of elements in it, counting multiplicity. Given a set  $G = \{g_1, \dots, g_k\} \subseteq F_n$ , together with a multi-set  $I = \{i_1, \dots, i_q\} \subseteq [k]$  of indices, let  $g_I$  denote the function such that  $g_I(x) = (g_{i_1}(x), \dots, g_{i_q}(x))$  for  $x \in \{0, 1\}^n$ . We say that two functions  $f$  and  $g$  in  $F_n$  are  $\delta$ -close if  $\Pr_{x \in \mathcal{U}_n}[f(x) \neq g(x)] \leq \delta$ . All the logarithms in this paper will have base two.

We will need the following simple lower bound on the tail probability of binomial distribution. We omit the proof due to the space constraint.

**Fact 1.** *Let  $Z_1, \dots, Z_t$  be i.i.d. binary random variables, with  $\mathbb{E}[Z_i] = \mu$  for every  $i \in [t]$ . Suppose  $\varepsilon < \frac{1}{5}$  and  $t = \Omega(\frac{1}{\varepsilon^2})$ . Then we have the following: (1) if  $\mu \leq \frac{1+2\varepsilon}{2}$ , then  $\Pr[\sum_{i \in [t]} Z_i \leq \frac{1-\varepsilon}{2}t] \geq 2^{-O(\varepsilon^2 t)}$ , and (2) if  $\mu \geq \frac{1-2\varepsilon}{2}$ , then  $\Pr[\sum_{i \in [t]} Z_i \geq \frac{1+\varepsilon}{2}t] \geq 2^{-O(\varepsilon^2 t)}$ .*

We will also need the following result, known as Turán’s Theorem, which can be found in standard textbooks (e.g. [1]).

**Fact 2.** *(Turán’s Theorem) Given a graph  $G = (V, E)$ , let  $d_v$  denote the degree of a vertex  $v$ . Then the size of its maximum independent set is at least  $\sum_{v \in V} \frac{1}{d_v+1}$ .*

### 2.1 Hardness and Hard-Core Set Lemma

We say that a function  $f \in F_n$  is  $\delta$ -hard (or has hardness  $\delta$ ) for size  $s$ , if for any  $C \in \text{SIZE}(s)$ ,  $\Pr_{x \in \mathcal{U}_n}[C(x) \neq f(x)] > \delta$ . Impagliazzo introduced the following notion of a hard-core set of a hard function.

**Definition 1.** [7] *We say that a function  $f \in F_n$  has an  $\varepsilon$ -hard-core set  $H \subseteq \{0, 1\}^n$  for size  $s$ , if for any  $C \in \text{SIZE}(s)$ ,  $\Pr_{x \in H}[C(x) \neq f(x)] > \frac{1-\varepsilon}{2}$ .*

Now we can state Impagliazzo’s hard-core set lemma [7], which is the focus of our paper.

**Lemma 1.** [7] *Any function  $f \in F_n$  which is  $\delta$ -hard for size  $s$  must have an  $\varepsilon$ -hard-core set  $H$  for size  $s'$ , with  $|H| \geq \delta 2^n$  and  $s' = O(s/(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}))$ .*

Note that in this lemma, the hardness on the set  $H$  is measured against a smaller circuit size  $s'$ , as compared to the original circuit size  $s$ . This was later improved by Klivans and Servedio [9] to  $s' = O(s/(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}))$  but at the expense of having a slightly smaller hard-core set of size  $\delta 2^{n-1}$ . A closer look at their proofs shows that they work for the more general setting with hardness measured against any class of functions instead of just circuits. For this, let us first formalize the notion that a function has no hard-core set for a class  $G \subseteq F_n$ .

**Definition 2.** Given a set  $G = \{g_1, \dots, g_k\} \subseteq F_n$ , we say that a function  $f \in F_n$  is  $(\delta, \varepsilon, G)$ -easy if for any  $H \subseteq \{0, 1\}^n$  of size  $\delta 2^n$ , there is a function  $g \in G$  such that  $\Pr_{x \in H} [g(x) \neq f(x)] \leq \frac{1-\varepsilon}{2}$ .

Then from [7] and its improvement in [9], one actually has the following.

**Lemma 2.** For some  $q = O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ , there exists a function  $D : \{0, 1\}^q \rightarrow \{0, 1\} \in \text{SIZE}(\text{poly}(q))$  such that for some constant  $c$  the following holds. For any  $G = \{g_1, \dots, g_k\} \subseteq F_n$ , if a function  $f \in F_n$  is  $(c\delta, \varepsilon, G)$ -easy, then there is a multi-set  $I$  with  $|I| = q$  such that  $\Pr_x [D(g_I(x)) \neq f(x)] \leq \delta$ .

In [7],  $c = 1$  and  $D$  is the majority function (and  $q = O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta\varepsilon})$ ), while in [9],  $c = 1/2$  and  $D$  is the majority of majority functions.

### 2.2 Black-Box Constructions of Hard-Core Sets

Now we introduce our two models for black-box construction of hard-core set. The first one is stronger than the second.

**Definition 3.** We say that a (non-uniform) oracle algorithm  $\text{DEC}^{(\cdot)}$  realizes a strongly black-box  $(\delta, \varepsilon, k)$ -construction (of a hard-core set) if for some  $q \in \mathbb{N}$  it has a decision function  $D : \{0, 1\}^q \rightarrow \{0, 1\}$  such that for some constant  $c$  the following holds. For any  $G = \{g_1, \dots, g_k\} \subseteq F_n$ , if a function  $f \in F_n$  is  $(c\delta, \varepsilon, G)$ -easy, then there is a multi-set  $I$  with  $|I| = q$  such that  $\text{DEC}^{G,I}(x) = D(g_I(x))$  and  $\Pr_x [\text{DEC}^{G,I}(x) \neq f(x)] \leq \delta$ . We call  $q$  the query complexity of  $\text{DEC}$ .

In this model,  $I$  can be seen as an advice, so the advice is of the form of a multi-set  $I = \{i_1, \dots, i_q\}$ , and the algorithm  $\text{DEC}$  is restricted to be of the following form: on input  $x$ , it queries the functions  $g_{i_1}, \dots, g_{i_q}$  all on the input  $x$ , applies the function  $D$  on the  $q$  answer bits, and outputs  $D(g_{i_1}(x), \dots, g_{i_q}(x))$ . Note that the known hard-core set constructions are in fact done in our first model [7,9]. Our second model generalizes the first one by removing the format constraint on the algorithm  $\text{DEC}$  and its advice. That is, the algorithm  $\text{DEC}$  and its advice now are allowed to be of arbitrary form.

**Definition 4.** We say that a (non-uniform) oracle algorithm  $\text{DEC}^{(\cdot)}$  realizes a weakly black-box  $(\delta, \varepsilon, k)$ -construction (of a hard-core set) if for some constant  $c$  the following holds. For any  $G = \{g_1, \dots, g_k\} \subseteq F_n$ , if a function  $f \in F_n$  is  $(c\delta, \varepsilon, G)$ -easy, then there is an advice string  $\alpha$  such that  $\Pr_x [\text{DEC}^{G,\alpha}(x) \neq f(x)] \leq \delta$ .

Note that in the two definitions above, we do not place any constraint on the computational complexity of  $\text{DEC}$ . Our first two results show that even having an unbounded computational power,  $\text{DEC}$  still needs to make a sufficient number of queries and use a sufficient number of advice bits in these two models, respectively. On the other hand, our third result targets on the computational complexity of  $\text{DEC}$  and show that it cannot be implemented in a low-level complexity class such as  $\text{AC}^0[p]$ . Here, we say that the oracle algorithm  $\text{DEC}^G$  can be implemented in a circuit class if the function  $\text{DEC}^G$  can be computed by a circuit in the class equipped with functions from  $G$  as oracle gates.

### 3 Query Complexity in Strongly Black-Box Constructions

In this section, we give a lower bound on the query complexity of any strongly black-box construction of hard-core set. Formally, we have the following.

**Theorem 1.** *Suppose  $2^{-c_1 n} \leq \varepsilon, \delta < c_2$ , and  $\omega(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}) \leq k \leq 2^{2^{c_3 n}}$ , for small enough constants  $c_1, c_2, c_3 > 0$ . Then any strongly black-box  $(\delta, \varepsilon, k)$ -construction must have a query complexity of  $q = \Omega(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ .*

Our lower bound is optimal since it is matched (up to a constant factor) by the upper bound from the construction of Klivans and Servedio (Lemma 2). Note that the assumption  $k \geq \omega(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}) \geq 2^{\Omega(n)}$  is reasonable, since in standard settings of the hard-core set lemma,  $G$  typically consists of circuits of polynomial (or larger) size, which gives  $k = |G| \geq 2^{\text{poly}(n)}$ .

The roadmap for the proof is the following. Consider any DEC which realizes such a strongly black-box construction. We would like to show the existence of a function  $f$  and a family  $G = \{g_1, \dots, g_k\}$  of functions such that  $f$  is  $(c\delta, \varepsilon, G)$ -easy (with  $c$  being the constant associated with DEC) but the algorithm DEC without making enough queries cannot approximate  $f$  well. We will prove their existence by a probabilistic argument.

Now we proceed to the proof of the theorem. Suppose the parameters  $\varepsilon, \delta, k$  satisfy the condition stated in the theorem. Suppose we have such a black-box construction realized by an oracle algorithm DEC with the decision function  $D$ . Consider  $k$  independent random functions  $b_1, \dots, b_k$  from  $F_n$ , which will serve as noise vectors, such that for any  $i$  and  $x$ ,  $\Pr[b_i(x) = 0] = \frac{1+2\varepsilon}{2}$ .

Now let  $f$  be a perfectly random function from  $F_n$ , so that  $\Pr[f(x) = 1] = \frac{1}{2}$  for any  $x$ , and let  $g_1, \dots, g_k$  be  $k$  independent noisy versions of  $f$  defined as  $g_i(x) = f(x) \oplus b_i(x)$ , for any  $i$  and  $x$ . Let  $B = \{b_1, \dots, b_k\}$  and  $G = \{g_1, \dots, g_k\}$ . First, we have the following, the proof of which is a simple application of a Chernoff bound and a union bound and is omitted due to the space constraint.

**Lemma 3.** *If  $k = \omega(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ , then  $\Pr_{f,G}[f \text{ is not } (c\delta, \varepsilon, G)\text{-easy}] = o(1)$ .*

Next, we show that with a small  $q$ , DEC is unlikely to approximate  $f$  well. Recall that for a multi-set  $I = \{i_1, \dots, i_q\} \subseteq [k]$ ,  $g_I(x) = (g_{i_1}(x), \dots, g_{i_q}(x))$ . We say that DEC can  $\delta$ -approximate  $f$  if there is a multi-set  $I \subseteq [k]$  with  $|I| = q$  such that  $D \circ g_I$  is  $\delta$ -close to  $f$  (i.e.,  $\Pr_x[D(g_I(x)) \neq f(x)] \leq \delta$ ).

**Lemma 4.** *If  $q = o(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ , then  $\Pr_{f,G}[\text{DEC can } \delta\text{-approximate } f] = o(1)$ .*

*Proof.* Consider any multi-set  $I \subseteq [k]$  with  $|I| = q$ . First we show the following.

*Claim.* For any  $x \in \{0, 1\}^n$ ,  $\Pr_{f,G}[D(g_I(x)) \neq f(x)] \geq 2\delta$ .

*Proof.* Let  $\tilde{I}$  denote the set of elements from  $I$ , removing multiplicity, and  $\tilde{D}$  the function such that  $\tilde{D}(g_{\tilde{I}}(x)) = D(g_I(x))$ . For example, for  $I = \{1, 1, 2\}$ , we have  $\tilde{I} = \{1, 2\}$  and  $\tilde{D}(g_1(x), g_2(x)) = D(g_1(x), g_1(x), g_2(x))$ . Then

$$\Pr_{f,G}[D(g_I(x)) \neq f(x)] = \Pr_{f,G}[\tilde{D}(g_{\tilde{I}}(x)) \neq f(x)] = \frac{1}{2}p(0) + \frac{1}{2}p(1),$$

where  $p(v) = \Pr_{f,G}[\tilde{D}(g_{\tilde{I}}(x)) = v \mid f(x) = 1 - v]$  for  $v \in \{0, 1\}$ , so it suffices to give a lower bound for either  $p(0)$  or  $p(1)$ . Let  $\tilde{I} = \{i_1, \dots, i_{\tilde{q}}\}$ , where  $\tilde{q}$  is clearly at most  $q$ . Assume without loss of generality that  $|\tilde{D}^{-1}(1)| \geq 2^{\tilde{q}-1}$ , and we will give a lower bound for  $p(1)$  (otherwise, we bound  $p(0)$  in a similar way). Let  $Z = (Z_1, \dots, Z_{\tilde{q}})$  denote the sequence of random variables  $(b_{i_1}(x), \dots, b_{i_{\tilde{q}}}(x))$ , which are i.i.d. with  $\mathbb{E}[Z_i] = \frac{1-2\varepsilon}{2}$ . Note that  $g_i(x) = b_i(x)$  when  $f(x) = 0$ , so

$$p(1) = \Pr_B \left[ \tilde{D}(b_{i_1}(x), \dots, b_{i_{\tilde{q}}}(x)) = 1 \right] = \sum_{y \in \tilde{D}^{-1}(1)} \Pr[Z = y].$$

The above is the sum of  $|\tilde{D}^{-1}(1)| \geq 2^{\tilde{q}-1}$  values from the  $2^{\tilde{q}}$  values:  $\Pr[Z = y]$  for  $y \in \{0, 1\}^{\tilde{q}}$ , so it is clearly no less than the sum of the  $2^{\tilde{q}-1}$  smallest values from them. Observe that  $\Pr[Z = y] = \left(\frac{1-2\varepsilon}{2}\right)^{\#_1(y)} \left(\frac{1+2\varepsilon}{2}\right)^{\tilde{q}-\#_1(y)}$ , where  $\#_1(y)$  denotes the number of 1's in the string  $y$ , so  $\Pr[Z = y] \leq \Pr[Z = y']$  whenever  $\#_1(y) \geq \#_1(y')$ . As a result,  $p(1)$  is at least

$$\sum_{y: \#_1(y) > \frac{1}{2}\tilde{q}} \Pr[Z = y] = \Pr \left[ \sum_{i \in [\tilde{q}]} Z_i > \frac{1}{2}\tilde{q} \right] \geq \Pr \left[ \sum_{i \in [\tilde{q}]} Z_i > \frac{1+\varepsilon}{2}\tilde{q} \right] \geq 2^{-O(\varepsilon^2\tilde{q})},$$

by Fact [2](#). So when  $\tilde{q} \leq q = o(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ , we have  $\Pr_{f,G} [D(g_I(x)) \neq f(x)] \geq \frac{1}{2}p(1) \geq 2\delta$ .  $\square$

Now for any multi-set  $I$  with  $|I| = q$ , let  $Y_x$ , for  $x \in \{0, 1\}^n$ , denote the binary random variable such that  $Y_x = 1$  if and only if  $D(g_I(x)) \neq f(x)$ . Clearly, they are i.i.d. random variables, and we know from above that  $\mathbb{E}[Y_x] \geq 2\delta$  for any  $x$ . So by Chernoff bound,  $\Pr_{f,G} [D \circ g_I \text{ is } \delta\text{-close to } f] = \Pr[\sum_x Y_x \leq \delta 2^n] \leq 2^{-\Omega(\delta^2 2^n)}$ . Then a union bound gives

$$\Pr_{f,G} [\exists I \text{ with } |I| = q : D \circ g_I \text{ is } \delta\text{-close to } f] \leq k^q \cdot 2^{-\Omega(\delta^2 2^n)} \leq o(1),$$

since  $\varepsilon, \delta \geq 2^{-c_1 n}$  and  $k \leq 2^{2^{c_3 n}}$ , for small enough constants  $c_1, c_3 > 0$ .  $\square$

From Lemma [3](#) and [4](#), we conclude that there exist  $f \in F_n$  and  $G = \{g_1, \dots, g_k\} \subseteq F_n$  which satisfy the following:

- $f$  is  $(c\delta, \varepsilon, G)$ -easy, and
- for every multi-set  $I \subseteq [k]$  of size  $q = o(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ ,  $\Pr_x [D(g_I(x)) \neq f(x)] > \delta$ .

Therefore, any DEC realizing a strongly black-box  $(\delta, \varepsilon, k)$ -construction must have  $q = \Omega(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ , when  $k = \omega(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ . This proves Theorem [1](#).

## 4 Advice Complexity in Weakly Black-Box Constructions

In this section, we show a length lower bound on the advice needed in any weakly black-box construction of hard-core set. This explains why a uniform version of the hard-core set lemma is hard to come by and any black-box construction is inherently non-uniform. Formally, we have the following.

**Theorem 2.** *Suppose  $2^{-c_1 n} \leq \varepsilon, \delta < c_2$ , and  $\frac{1}{\varepsilon^3} \leq k \leq 2^{2^{c_3 n}}$ , for small enough constants  $c_1, c_2, c_3 > 0$ . Then any weakly black-box  $(\delta, \varepsilon, k)$ -construction must need an advice of length  $\Omega(\frac{1}{\varepsilon} \log k)$ .*

As a comparison, the construction of Klivans and Servedio (Lemma 2) provides an upper bound of  $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta} \log k)$  on the advice length, so there is a gap of a factor  $O(\frac{1}{\varepsilon} \log \frac{1}{\delta})$  between our lower bound and their upper bound. As in Theorem 1, one can also argue that the range assumption on the parameters is reasonable.

The roadmap for the proof is the following. Consider any DEC realizing such a weakly black-box construction. We will show the existence of a family  $G = \{g_1, \dots, g_k\} \subseteq F_n$  with respect to which we can find a large collection  $\Gamma$  of functions satisfying the following two properties: (1) any function in  $\Gamma$  is  $(c\delta, \varepsilon, G)$ -easy (with  $c$  the constant associated with DEC), and (2) any two functions in  $\Gamma$  are not  $2\delta$ -close. This then implies a lower bound of  $\log |\Gamma|$  on the advice length. Again, we will show the existence by a probabilistic argument.

Now we proceed to the proof of the theorem. First, we independently sample  $k$  perfectly random functions  $g_1, \dots, g_k \in F_n$  (for any  $i$  and  $x$ ,  $g_i(x) = 1$  with probability exactly  $\frac{1}{2}$ ), and let  $G = \{g_1, \dots, g_k\}$ . Now for any set  $I = \{i_1, \dots, i_t\} \subseteq [k]$ , let  $G_I$  be the function such that  $G_I(x) = \text{MAJ}(g_{i_1}(x), \dots, g_{i_t}(x))$ , where MAJ denotes the majority function. Then we have the following, which follows from the known result that any majority gate has a good correlation with one of its input bits [43].

**Lemma 5.** *Let  $G = \{g_1, \dots, g_k\}$  be any set of functions from  $F_n$ . Then for any  $I \subseteq [k]$ , the function  $G_I$  is  $(c\delta, \frac{1}{|I|}, G)$ -easy.*

Let  $t = \lfloor \frac{1}{\varepsilon} \rfloor$ , let  $V^t = \{I \subseteq [k] : |I| = t\}$ , and consider the class  $\{G_I : I \in V^t\}$  of functions. Lemma 5 tells us that each function in the class is  $(c\delta, \varepsilon, G)$ -easy. Our next step is to find a large collection of functions from this class such that any two of them are not close. Note that whether or not two functions  $G_I, G_J$  are close really depends on the choice of  $G$ . We will show that if  $I$  and  $J$  have a small intersection, then  $G_I$  and  $G_J$  are unlikely to be close for a random  $G$ .

**Lemma 6.**  $\forall I, J \in V^t$  with  $|I \cap J| < \frac{t}{2}$ ,  $\Pr_G [G_I \text{ is } 2\delta\text{-close to } G_J] \leq 2^{-\Omega(2^n)}$ .

*Proof.* Consider any such  $I$  and  $J$ . First, we prove the following.

*Claim.* For any  $x \in \{0, 1\}^n$ ,  $\Pr_G [G_I(x) \neq G_J(x)] = \Omega(1)$ .

*Proof.* Note that for any  $x$ ,  $g_1(x), \dots, g_k(x)$  can be seen as a sequence of i.i.d. binary random variables  $Z_1, \dots, Z_k$ , with  $E[Z_i] = \frac{1}{2}$  for each  $i$ . Let  $Z_I$  denote the subsequence of random variables  $Z_i$  for  $i \in I$ , and similarly for  $Z_J$ . Thus our goal is to show that  $\Pr[\text{MAJ}(Z_I) \neq \text{MAJ}(Z_J)] = \Omega(1)$ .

Let  $K = I \cap J$ ,  $I_1 = I \setminus K$ , and  $J_1 = J \setminus K$ , and note that  $|K| < |I_1|, |J_1|$ . Consider the following three events.

- $A_1$ :  $\left| \sum_{i \in K} Z_i - \frac{|K|}{2} \right| \leq \frac{1}{2} \sqrt{|K|}$ . By Chernoff bound,  $\Pr[\neg A_1] < \alpha$  for a constant  $\alpha < 1$ , so  $\Pr[A_1] = \Omega(1)$ .



- $A_2$ :  $\sum_{i \in I_1} Z_i \leq \frac{1}{2}(|I_1| - \sqrt{|I_1|})$ . By Fact □ (1) with  $\mu = \frac{1}{2}$ ,  $\Pr[A_2] = \Omega(1)$ .
- $A_3$ :  $\sum_{i \in J_1} Z_i \geq \frac{1}{2}(|J_1| + \sqrt{|J_1|})$ . By Fact □ (2) with  $\mu = \frac{1}{2}$ ,  $\Pr[A_3] = \Omega(1)$ .

Now observe that if  $A_1 \wedge A_2$ , then

$$\sum_{i \in I} Z_i \leq \frac{1}{2}(|K| + |I_1| + \sqrt{|K|} - \sqrt{|I_1|}) < \frac{|K| + |I_1|}{2} = \frac{|I|}{2},$$

which implies that  $\text{MAJ}(Z_I) = 0$ . Similarly, if  $A_1 \wedge A_3$ , then

$$\sum_{i \in J} Z_i \geq \frac{1}{2}(|K| + |J_1| - \sqrt{|K|} + \sqrt{|J_1|}) > \frac{|K| + |J_1|}{2} = \frac{|J|}{2},$$

which implies that  $\text{MAJ}(Z_J) = 1$ . That is, if  $A_1 \wedge A_2 \wedge A_3$ , then  $\text{MAJ}(Z_I) = 0 \wedge \text{MAJ}(Z_J) = 1$ , so  $\text{MAJ}(Z_I) \neq \text{MAJ}(Z_J)$ . Since the events  $A_1, A_2, A_3$  are independent from each other (as each depends on a separate set of random variables), we have

$$\Pr[\text{MAJ}(Z_I) \neq \text{MAJ}(Z_J)] \geq \Pr[A_1 \wedge A_2 \wedge A_3] = \Pr[A_1] \cdot \Pr[A_2] \cdot \Pr[A_3] \geq \Omega(1).$$

□

From this, we next show that  $G_I$  and  $G_J$  are unlikely to agree on many  $x$ . For any  $x \in \{0, 1\}^n$ , consider the binary random variable  $Y_x$  such that  $Y_x = 1$  if and only if  $G_I(x) \neq G_J(x)$ . From the above claim, we know that  $\mathbb{E}_G[Y_x] \geq c_0$  for some constant  $c_0$ . So by Chernoff bound, we have

$$\Pr_G[G_I \text{ is } 2\delta\text{-close to } G_J] = \Pr\left[\sum_x Y_x \leq 2\delta 2^n\right] \leq 2^{-\Omega((c_0 - 2\delta)^2 2^n)} \leq 2^{-\Omega(2^n)},$$

as we assume that  $\delta < c_2$  for a small enough constant  $c_2$ . □

Call  $G$  nice if for any  $I, J \in V^t$  with  $|I \cap J| < \frac{t}{2}$ ,  $G_I$  is not  $2\delta$ -close to  $G_J$ . By a union bound,  $\Pr_G[G \text{ is not nice}] \leq \binom{k}{t}^2 \cdot 2^{-\Omega(2^n)} \leq 2^{2t \log k} \cdot 2^{-\Omega(2^n)} < 1$ , as  $t \leq \frac{1}{\epsilon} \leq 2^{c_1 n}$  and  $k \leq 2^{2^{c_3 n}}$ , for small enough constants  $c_1, c_3 > 0$ . This guarantees the existence of a nice  $G$ . From now on, we will fix on one such  $G$ .

Consider the undirected graph  $\mathcal{G} = (V, E)$  where  $V = \{G_I : I \in V^t\}$  and  $E$  consists of those pairs of  $G_I, G_J$  which are  $2\delta$ -close to each other. Then we have the following.

**Lemma 7.**  $\mathcal{G}$  has an independent set of size at least  $k^{\Omega(t)}$ .

*Proof.* Since  $G$  is nice, there cannot be an edge between vertices  $G_I$  and  $G_J$  if  $|I \cap J| < \frac{t}{2}$ . Thus, the degree of any vertex  $G_I$  is at most the number of  $G_J$  with  $|I \cap J| \geq \frac{t}{2}$ , which is at most

$$\sum_{\frac{t}{2} \leq i < t} \binom{t}{i} \binom{k-t}{t-i} \leq \sum_{\frac{t}{2} \leq i < t} \binom{t}{i} \binom{k}{\frac{t}{2}} \leq 2^t \binom{k}{\frac{t}{2}} \leq \left(\frac{8ek}{t}\right)^{t/2} \leq k^{t/2},$$

where the first and last inequalities, respectively, hold under the conditions that  $k \geq \frac{1}{\varepsilon^3} \geq t^3$  and  $t = \lfloor \frac{1}{\varepsilon} \rfloor$  is at least a large enough constant, while the third inequality uses the fact that  $\binom{n}{m} \leq (\frac{en}{m})^m$ . Then by Turán’s theorem (Fact 2),  $\mathcal{G}$  has an independent set of size  $\binom{k}{t} \frac{1}{k^{t/2+1}} \geq (\frac{k}{t})^t \frac{1}{k^{t/2+1}} \geq k^{2t/3} \frac{1}{k^{t/2+1}} \geq k^{\Omega(t)}$ , where the second inequality follows from the assumption that  $k \geq t^3$ .  $\square$

Now we are ready to finish the proof of the theorem. From Lemma 7 we know that  $\mathcal{G}$  has an independent set  $\Gamma$  of size  $k^{\Omega(t)}$ . Note that any two  $G_I, G_J \in \Gamma$  are not  $2\delta$ -close. Furthermore, we know from Lemma 5 that every  $G_I \in \Gamma$  is  $(c\delta, \varepsilon, G)$ -easy, since  $|I| = t \leq \frac{1}{\varepsilon}$ . Therefore, an advice of length  $\log |\Gamma| = \Omega(t \log k) = \Omega(\frac{1}{\varepsilon} \log k)$  is required, because for each advice  $\alpha$ ,  $\text{DEC}^{G,\alpha}$  can only be  $\delta$ -close to at most one  $G_I \in \Gamma$ . This proves Theorem 2.

### 5 No Weakly Black-Box Construction in $\text{AC}^0[p]$

In this section, we show that no weakly black-box construction of hard-core set can be implemented in  $\text{AC}^0[p]$ . More precisely we have the following.

**Theorem 3.** *Suppose  $1 < \delta < 1/20$ ,  $0 < \varepsilon < 1$ ,  $k \geq n$ , and  $p$  a prime. Let  $t = \min(\lfloor 1/\varepsilon \rfloor, n)$ . Then no weakly black-box  $(\delta, \varepsilon, k)$ -construction can be implemented in  $\text{AC}^0[p](2^{\text{poly}(\log t)})$ .*

The idea is the following. Suppose we have a function  $\text{DEC}$  realizing such a black-box construction. Let  $I = [t]$  and note that  $1/t \geq \varepsilon$ . From the previous section, we know that for any  $G$ , the function  $G_I$  is  $(c\delta, \varepsilon, G)$ -easy (with  $c$  the constant associated with  $\text{DEC}$ ), so there must exist some advice  $\alpha$  such that  $\text{DEC}^{G,\alpha}$  is  $\delta$ -close to the function  $G_I$ , which is the majority function over  $g_1, \dots, g_t$ . As will be shown later, by defining  $G$  properly, we can use  $\text{DEC}^{G,\alpha}$  to approximate the majority function on  $t$  input variables. Then we need the following lower bound on the majority function. The proof is based on bounds from [13,14,15] and is omitted due to the space constraint.

**Lemma 8.** *For any  $C : \{0, 1\}^t \rightarrow \{0, 1\}$  in  $\text{AC}^0[p](2^{\text{poly}(\log t)})$  and for a large enough  $t$ , we have  $\Pr_x [C(x) \neq \text{MAJ}(x)] \geq 1/20$ .*

We define the function  $g_i$  as  $g_i(x) = x_i$  for  $i \in [n]$  and  $g_i(x) = 0$  otherwise, for  $x \in \{0, 1\}^n$ . Let  $G = \{g_1, \dots, g_k\}$ . Then  $G_I(x) = \text{MAJ}(x_1, \dots, x_t)$  for any  $x \in \{0, 1\}^n$ , so there must be some advice  $\alpha$  such that  $\Pr_x [\text{DEC}^{G,\alpha}(x) \neq \text{MAJ}(x_1, \dots, x_t)] \leq \delta$ , and by an average argument there must be some fixed  $\bar{x}_{t+1}, \dots, \bar{x}_n$  such that

$$\Pr_{x_1, \dots, x_t} [\text{DEC}^{G,\alpha}(x_1, \dots, x_t, \bar{x}_{t+1}, \dots, \bar{x}_n) \neq \text{MAJ}(x_1, \dots, x_t)] \leq \delta.$$

Such  $\alpha$  and  $\bar{x}_{t+1}, \dots, \bar{x}_n$  can be hard-wired into the circuit for  $\text{DEC}$ , and observe that all the oracle gates can be removed as every oracle query can be answered by some input bit  $x_i$  or a fixed constant. So if  $\text{DEC}^G$  can be computed by an  $\text{AC}^0[p](2^{\text{poly}(\log t)})$  circuit equipped with oracle gates from  $G$ , we can obtain from it an  $\text{AC}^0[p](2^{\text{poly}(\log t)})$  circuit (without oracle gates) which is  $\delta$ -close to the majority function on  $t$  bits and contradicts Lemma 8. This proves Theorem 3.

## References

1. Alon, N., Spencer, J.: The probabilistic method, 2nd edn. Wiley-Interscience, New York (2000)
2. Babai, L., Fortnow, L., Nisan, N., Wigderson, A.: BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity* 3(4), 307–318 (1993)
3. Goldmann, M., Håstad, J., Razborov, A.: Majority gates vs. general weighted threshold gates. *Computational Complexity* 2, 277–300 (1992)
4. Hajnal, A., Maass, W., Pudlák, P., Szegedy, M., Turán, G.: Threshold circuits of bounded depth. In: *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*, pp. 99–110 (1987)
5. Healy, A., Vadhan, S., Viola, E.: Using nondeterminism to amplify hardness. *SIAM Journal on Computing* 35(4), 903–931 (2006)
6. Holenstein, T.: Key agreement from weak bit agreement. In: *Proceedings of the 37th ACM Symposium on Theory of Computing*, pp. 664–673. ACM Press, New York (2005)
7. Impagliazzo, R.: Hard-core distributions for somewhat hard problems. In: *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pp. 538–545. IEEE Computer Society Press, Los Alamitos (1995)
8. Impagliazzo, R., Wigderson, A.:  $P=BPP$  if  $E$  requires exponential circuits: Derandomizing the XOR lemma. In: *Proceedings of the 29th ACM Symposium on Theory of Computing*, pp. 220–229. ACM Press, New York (1997)
9. Klivans, A., Servadio, R.A.: Boosting and hard-core sets. *Machine Learning* 51(3), 217–238 (2003)
10. Lu, C.-J., Tsai, S.-C., Wu, H.-L.: On the complexity of hardness amplification. In: *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pp. 170–182. IEEE Computer Society Press, Los Alamitos (2005)
11. O’Donnell, R.: Hardness amplification within NP. In: *Proceedings of the 34th ACM Symposium on Theory of Computing*, pp. 751–760. ACM Press, New York (2002)
12. Sudan, M., Trevisan, L., Vadhan, S.: Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences* 62(2), 236–266 (2001)
13. Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: *Proceedings of the 19th ACM Symposium on Theory of Computing*, pp. 77–82. ACM Press, New York (1987)
14. Szegedy, M.: Algebraic methods in lower bounds for computational models with limited communication. Ph.D. thesis, University of Chicago (1989)
15. Tarui, J.: Degree complexity of boolean functions and its applications to relativized separations. In: *Proceedings of the 6th Annual IEEE Conference on Structure in Complexity Theory*, pp. 382–390. IEEE Computer Society Press, Los Alamitos (1991)
16. Trevisan, L.: List decoding using the XOR lemma. In: *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pp. 126–135. IEEE Computer Society Press, Los Alamitos (2003)
17. Trevisan, L.: On uniform amplification of hardness in NP. In: *Proceedings of the 37th ACM Symposium on Theory of Computing*, pp. 31–38. ACM Press, New York (2005)
18. Viola, E.: The Complexity of Hardness Amplification and Derandomization. Ph.D. thesis, Harvard University (2006)
19. Yao, A.: Theory and applications of trapdoor functions. In: *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 80–91. IEEE Computer Society Press, Los Alamitos (1982)

# Approximation by DNF: Examples and Counterexamples

Ryan O'Donnell and Karl Wimmer

Carnegie Mellon University, Pittsburgh PA 15213, USA  
odonnell@cs.cmu.edu, kwimmer@andrew.cmu.edu

**Abstract.** Say that  $f : \{0, 1\}^n \rightarrow \{0, 1\}$   $\epsilon$ -approximates  $g : \{0, 1\}^n \rightarrow \{0, 1\}$  if the functions disagree on at most an  $\epsilon$  fraction of points. This paper contains two results about approximation by DNF and other small-depth circuits:

(1) For every constant  $0 < \epsilon < 1/2$  there is a DNF of size  $2^{O(\sqrt{n})}$  that  $\epsilon$ -approximates the Majority function on  $n$  bits, and this is optimal up to the constant in the exponent.

(2) There is a monotone function  $\mathcal{F} : \{0, 1\}^n \rightarrow \{0, 1\}$  with total influence (AKA average sensitivity)  $\mathbb{I}(\mathcal{F}) \leq O(\log n)$  such that any DNF or CNF that  $.01$ -approximates  $\mathcal{F}$  requires size  $2^{\Omega(n/\log n)}$  and such that *any* unbounded fan-in AND-OR-NOT circuit that  $.01$ -approximates  $\mathcal{F}$  requires size  $\Omega(n/\log n)$ . This disproves a conjecture of Benjamini, Kalai, and Schramm (appearing in [\[BKS99\]](#), [\[Kal00\]](#), [\[KS05\]](#)).

## 1 Introduction

### 1.1 Definitions

This paper is concerned with approximating boolean functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  by DNF formulas of small size. Let us first give the requisite definitions.

*Circuits:* We will consider single-output **circuits** composed of unbounded fan-in AND and OR gates over the input **literals** (inputs and negated inputs). The **size** of a circuit is the number of AND and OR gates it contains, and the **depth** of the circuit is the number of AND and OR gates on the longest path from an input bit to the output gate. We will also make the not completely standard definition that the **width** of a circuit is the maximum, over all AND and OR gates, of the number of *literals* feeding into the gate.

We will only be concerned with constant-depth circuits in this paper, and we will be particularly interested in depth 2. We assume circuits of depth 2 are always given by an OR of ANDs of literals, in which case they are **DNFs**, or by an AND of ORs of literals, in which case they are **CNFs**. The ORs in a DNF are called its **terms** and the ANDs in a CNF are called its **clauses**.

Finally, we will often identify a circuit over  $n$  input bits with the boolean function  $\{0, 1\}^n \rightarrow \{0, 1\}$  that it computes.

*Approximation:* Given two functions  $f, g : \{0, 1\}^n \rightarrow \{0, 1\}$ , we will say that  $f$   **$\epsilon$ -approximates**  $g$ , or  $f$  is an  **$\epsilon$ -approximator** for  $g$ , if the fraction of inputs in  $\{0, 1\}^n$  on which they disagree is at most  $\epsilon$ . We will also write this as

$$\Pr_{\mathbf{x}}[f(\mathbf{x}) \neq g(\mathbf{x})] \leq \epsilon,$$

with the convention that boldface letters are random variables, and that they are drawn from the uniform distribution on  $\{0, 1\}^n$  unless otherwise specified.

We will later need the following well known observation, showing that small-size circuits are well approximated by small-width circuits:

**Observation 1.** *If  $C$  is a circuit of size  $s$ , then for every  $\epsilon > 0$  there is a “simplification”  $C'$  of  $C$  that  $\epsilon$ -approximates  $C$  and has width at most  $\log(s/\epsilon)$ .<sup>1</sup> By “simplification” we mean that  $C'$  is obtained from  $C$  by replacing some of its gates with constants, so that  $C'$  has size and depth no more than  $C$ , and  $C'$  is a DNF (respectively, CNF) if  $C$  is.*

*Proof.* (Sketch.) Replace all AND gates with fan-in at least  $\log(s/\epsilon)$  by 0; replace all OR gates with fan-in at least  $\log(s/\epsilon)$  by 1. □

### 1.2 Approximation by DNF

DNF formulas are one of the simplest and most natural representation classes for boolean functions. Although every function can be computed by a DNF, some functions on  $n$  bits may require DNFs of size  $\Omega(2^n)$ . The natural question we pursue in this paper is whether this size can be significantly reduced for a given function if we are only required to  $\epsilon$ -approximate it, for some small constant  $\epsilon$ . Positive results along these lines would have interesting applications in several research areas, including computational learning theory and the study of threshold phenomena in random graphs; these topics will be discussed in Sections 1.3 and 1.4, respectively. However there do not seem to be many results on either upper or lower bounds for approximation by DNF in the literature.

A notable conjecture in this area was made 8 years ago by Benjamini, Kalai, and Schramm [BKS99] (published again in [Kal00, KS05]). To describe this conjecture, which we call the BKS Conjecture, we need to recall the notion of **total influence** [KKL88, LMN93]:

**Definition 1.** *Given a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , the influence of the  $i$ th coordinate on  $f$  is*

$$\text{Inf}_i(f) = \Pr_{\mathbf{x}}[f(\mathbf{x}) \neq f(\sigma_i \mathbf{x})],$$

where  $\sigma_i \mathbf{x}$  denotes  $\mathbf{x}$  with its  $i$ th bit flipped. The total influence (or average sensitivity) of  $f$  is

$$\mathbb{I}(f) = \sum_{i=1}^n \text{Inf}_i(f) = \mathbf{E}_{\mathbf{x}}[\#\{y \sim \mathbf{x} : f(y) \neq f(\mathbf{x})\}],$$

where the notation  $y \sim x$  means that the Hamming distance between  $y$  and  $x$  is 1.

---

<sup>1</sup> In this paper  $\log$  denotes  $\log_2$ .

The total influence is an important measure of the complexity of a function, used frequently in learning theory, threshold phenomena, and complexity theory. One important result to note is that constant-depth circuits of small size have small total influence:

**Theorem 2.** *Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be computed by a circuit of depth  $d$  and size  $s$ . Then  $\mathbb{I}(f) \leq O(\log^{d-1} s)$ .*

This was first proved by Boppana [Bop97], tightening an argument made by Linial, Mansour, and Nisan [LMN93] based on Håstad’s Switching Lemma [Hås86]. Note that the  $d = 2$  case of this theorem is quite easy, building on the simple result that  $\mathbb{I}(f) \leq 2w$  for any  $f$  computable by a DNF of width  $w$ .

We can now state Benjamini, Kalai, and Schramm’s conjecture, which essentially asserts a converse to Theorem 2 for monotone functions:

*BKS Conjecture: For every  $\epsilon > 0$  there is a constant  $K = K(\epsilon) < \infty$  such that the following holds: Every monotone  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  can be  $\epsilon$ -approximated by a depth- $d$  circuit of size at most*

$$\exp\left((K \cdot \mathbb{I}(f))^{1/(d-1)}\right),$$

for some  $d \geq 2$ .

(Recall that  $f$  is **monotone** if  $x \leq y \Rightarrow f(x) \leq f(y)$ .) Observation 1 implies that the BKS Conjecture could also add the condition that width is at most  $(K \cdot \mathbb{I}(f))^{1/(d-1)}$  without loss of generality.

If this conjecture were true it would be an important characterization of monotone functions with small total influence; if it were further true with  $d$  fixed to 2 it would yield very interesting positive results for approximation by DNF (or CNF).

### 1.3 Approximating Majority by DNF

Suppose the BKS Conjecture were true even with  $d$  fixed to 2. This would imply that for every constant  $\epsilon > 0$ , every monotone function  $f$  could be  $\epsilon$ -approximated by a DNF or CNF of size  $\exp(O(\mathbb{I}(f)))$ . Using Observation 1, we could further make the width of the approximator  $O(\mathbb{I}(f))$ . One reason to hope that this is true is that it *is* true, even for non-monotone functions, if one allows a more powerful class of depth-2 circuits:

**Definition 2.** A TOP (“threshold of parities” [Jac95]) is a depth-2 circuit with Parity gates at the lower level and a Majority gate on top.

**Proposition 1.** *For all  $\epsilon > 0$ , every boolean function  $f$  is  $\epsilon$ -approximated by a TOP of width  $O(\mathbb{I}(f)/\epsilon)$ .*

This proposition was shown in [KKL88, LMN93] by relating the total influence of a function to its Fourier spectrum.

TOP circuits arise frequently as the hypothesis class in many uniform-distribution learning algorithms. Examples include Linial, Mansour, and Nisan’s

algorithm for learning depth- $d$  size- $s$  circuits [LMN93], Jackson’s Harmonic Sieve algorithm for learning polynomial-size DNFs [Jac95], Bshouty and Tamon’s algorithm for learning monotone functions [BT96], and O’Donnell and Servedio’s algorithm for learning monotone polynomial-size decision trees [OS06]. (Incidentally, except for Jackson’s algorithm, all of these proceed by proving upper bounds on total influence.) An open question in learning theory is whether these algorithms (especially Jackson’s DNF algorithm) can be made to use the simpler class of DNFs as their hypothesis class.

This suggests the idea of trying to approximate TOPs by DNFs. By Proposition 1, approximating TOPs by DNFs could also be considered a way of attacking the BKS Conjecture. Now the Parities in a TOP could be converted to DNFs or CNFs of no greater width. But how to approximate the Majority by a small DNF or CNF is an interesting question. We solve the problem of  $\epsilon$ -approximating Majority by DNFs in Sections 2 and 3. Unfortunately, the size necessary is too large to give good approximations of TOPs.

The question of computing Majority by small circuits has a long and interesting history. Significant work has gone into computing Majority with small circuits of various sorts [PPZ92, AKS83, HMP06, Bop86, Val84]. Some of this work involves the subproblem of constructing small circuits for “approximate-Majority” — i.e., circuits that correctly compute Majority whenever the number of 1’s in the input is at least a  $2/3$  fraction or at most a  $1/3$  fraction. Note that this notion of approximation is not at all the same as our notion. Constructions of constant-depth circuits for this “approximate-Majority” have had important consequences for complexity theory [Ajt83, Ajt93, Vio05]. It seems, however, that no paper has previously investigated the existence of small constant-depth circuits for Majority that are  $\epsilon$ -approximators in our sense.

Our result on this topic is the following, following from the main results proved in Sections 2 and 3:

**Theorem 3.** *For every constant  $0 < \epsilon < 1/2$ , the Majority function on  $n$  bits can be  $\epsilon$ -approximated by a DNF of size  $\exp(O(\sqrt{n}))$ , and this is best possible up to the constant in the exponent.*

Note that the following fact is well known:

**Proposition 2.** *Every monotone function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  satisfies  $\mathbb{I}(f) \leq \mathbb{I}(\text{Maj}_n) = (\sqrt{2/\pi} + o(1))\sqrt{n}$ .*

Thus Theorem 3 shows that the BKS Conjecture with  $d$  fixed to 2 is true for the Majority function.

Our proof of the upper bound in Theorem 3 is by the probabilistic method; we essentially use the random DNF construction of Talagrand [Tal96]. Our proof of the lower bound in Theorem 3 uses the Kruskal-Katona Theorem to show that even  $\epsilon$ -approximators for Majority must have total influence  $\Omega(\sqrt{n})$ ; the lower bound then follows from Theorem 2:

**Theorem 4.** *Suppose  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is a  $(1/2 - \delta)$ -approximator for  $\text{Maj}_n$ , for constant  $\delta > 0$ . Then any depth- $d$  circuit computing  $f$  requires size  $\exp(\Omega(n^{1/(2d-2)}))$ .*

We remark that switching lemmas do not seem to provide any help in proving lower bounds on  $\epsilon$ -approximating DNFs; a short discussion of this appears in the full version of this paper.

### 1.4 Threshold Phenomena and the BKS Conjecture

One of the main motivations behind the BKS Conjecture is to provide general conditions under which a monotone function has large total influence. Benjamini, Kalai, and Schramm made their conjecture in the context of problems about threshold phenomena and noise sensitivity in random graphs. There, proving lower bounds on total influence is important, as the total influence relates to certain “critical exponents” in percolation problems, and it also captures the sharpness of “thresholds” for graph properties.

To understand the connection to threshold phenomena, consider the Erdős-Rényi random graph model on  $v$  vertices, and write  $n = \binom{v}{2}$ . Now a boolean string in  $\{0, 1\}^n$  can be identified with a graph, and a boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  can be identified with a collection of graphs. We say that  $f$  is a **graph property** if it closed under permutations of the  $v$  vertices. Suppose  $f$  is a nontrivial *monotone* graph property (i.e.,  $f$  is a monotone function that is not constantly 0 or 1). Then as we increase the edge probability  $p$  from 0 to 1, the probability that a random graph from the  $p$ -biased distribution on  $\{0, 1\}^n$  satisfies  $f$  increases continuously from 0 to 1. Hence there will be a critical exponent  $p^*$  where the probability of a random graph satisfying  $f$  is  $1/2$ . It is of great interest to understand how rapidly the probability of satisfying  $p$  jumps from near 0 to near 1 in the interval around  $p^*$ . The Russo-Margulis Lemma [Mar74, Rus78] shows that  $\frac{\partial}{\partial p} \mathbf{E}[f] = 4p(1-p)\mathbb{I}^{(p)}(f)$ , for an appropriate  $p$ -biased definition of total influence. It follows that graph properties having “sharp” thresholds corresponds to them having large total influence.

A celebrated theorem of Friedgut [Fri99] provides a version of the depth-2 BKS Conjecture in the context of graph properties with  $p^* = o(1)$ :

*Friedgut’s Theorem.* *There is a function  $K = K(C, \epsilon) < \infty$  such that the following holds: If  $f$  is a monotone graph property with critical probability  $p^* = o(1)$  and  $\mathbb{I}^{(p^*)}(f) \leq C$ , then  $f$  can be  $\epsilon$ -approximated (with respect to the  $p^*$ -biased distribution on  $\{0, 1\}^n$ ) by a DNF of width  $K(C, \epsilon)$ . In particular, one can take  $K(C, \epsilon) = O(C/\epsilon)$ .*

This result has been used with great success to show that various natural graph properties — and also random  $k$ -SAT problems — have sharp thresholds (see, e.g., [Fri05]); one proves this essentially by showing that the property cannot be well approximated by a small-width DNF.

The relationship between sharp thresholds and large total influence continues to hold in the context of general monotone boolean functions (i.e., not necessarily graph properties). Indeed, there has been significant interest in trying to extend Friedgut’s Theorem to the general, no-symmetry case. The BKS Conjecture is one proposal for such an extension (in the case of  $p^* = 1/2$ ). It is weaker than the Friedgut Theorem in that it allows for approximating circuits of depth greater



than 2. However the BKS Conjecture’s size/width bound for  $d = 2$  is very strong, essentially matching Friedgut’s Theorem — it states that in the  $d = 2$  case,  $K$  may be taken to have a linear dependence on  $\mathbb{I}(f)$ .

Some partial progress has been made towards proving Friedgut’s Theorem in the case of general monotone boolean functions. In an appendix to Friedgut’s paper, Bourgain [Fri99] showed that every boolean function with  $\mathbb{I}(f) \leq C$  has a Fourier coefficient  $\hat{f}(S)$  with  $|S| \leq O(C)$  and  $|\hat{f}(S)| \geq \exp(-O(C^2))$ ; he used this to show that when  $f$  is monotone and  $p^* = o(1)$ , there is a term of width  $O(C)$  that has  $\exp(-O(C^2))$ -correlation with  $f$ . Friedgut himself later showed [Fri98] that his theorem can be extended to general functions, even non-monotone ones (assuming  $p^*$  is bounded away from 0 and 1), at the expense of taking  $K(C, \epsilon) = \exp(O(C/\epsilon))$ .

However it turns out that these generalizations cannot be taken too far — our main result in Section 4 is that the BKS Conjecture is false. Specifically, we show:

**Theorem 5.** *There is a monotone function  $\mathcal{F} : \{0, 1\}^n \rightarrow \{0, 1\}$  with total influence  $\mathbb{I}(\mathcal{F}) \leq O(\log n)$  satisfying the following: Any DNF or CNF that .01-approximates  $\mathcal{F}$  requires width  $\Omega(n/\log n)$  and hence size  $2^{\Omega(n/\log n)}$ ; and, any circuit that .01-approximates  $\mathcal{F}$  requires size  $\Omega(n/\log n)$ .*

This rules out the BKS Conjecture. In particular, it shows that Friedgut’s Theorem cannot be proved for general monotone functions (in the  $p^* = 1/2$  case) unless one takes  $K(C, .01) \geq \exp(\Omega(C))$ . We remark that the function  $\mathcal{F}$  used in the theorem is computed by a polynomial-size, depth-3 circuit.

## 2 Approximating Majority

In this section we show how to construct a DNF of size  $2^{O(\sqrt{n}/\epsilon)}$  that  $\epsilon$ -approximates Majority on  $n$  bits. In the next section we explain why this result is optimal up to the constant in the exponent.

**Theorem 6.** *For all  $\epsilon \geq 1/\sqrt{n}$ , there is a DNF of width  $w = \frac{1}{\epsilon}\sqrt{n}$  and size  $(\ln 2)2^w$  which is an  $O(\epsilon)$ -approximator for  $\text{Maj}_n$ .*

*Proof.* Our construction is by the probabilistic method, inspired by the random DNF construction of Talagrand [Tal96]. Specifically, let  $\mathbf{D}$  be a randomly chosen DNF with  $(\ln 2)2^w$  terms, where each term is chosen by picking  $w$  variables independently with replacement. It suffices to show that

$$\mathbf{E}_{\mathbf{D}}[\Pr_{\mathbf{x}}[\mathbf{D}(\mathbf{x}) \neq \text{Maj}(\mathbf{x})]] \leq O(\epsilon), \quad (1)$$

because then a particular  $\mathbf{D}$  must exist which has  $\Pr_{\mathbf{x}}[\mathbf{D}(\mathbf{x}) \neq \text{Maj}(\mathbf{x})] \leq O(\epsilon)$ . Due to space limitations, the proof of (1) is deferred to the full version of this paper.  $\square$

### 3 A Lower Bound for Majority, Via Total Influence

The main result in this section shows that corrupting the Majority function,  $\text{Maj}_n$ , on even a large fraction of strings cannot decrease its total influence very much:

**Theorem 7.** *Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be an  $\epsilon$ -approximator for  $\text{Maj}_n$ . Then*

$$\mathbb{I}(f) \geq \begin{cases} (1 - O(\epsilon)) \cdot \mathbb{I}(\text{Maj}_n) & \text{if } \omega\left(\frac{1}{\sqrt{n}}\right) \leq \epsilon \leq 1/4, \\ \Omega(\epsilon) \cdot \mathbb{I}(\text{Maj}_n) & \text{if } 1/4 \leq \epsilon \leq 1/2 - \omega\left(\frac{1}{\sqrt{n}}\right). \end{cases}$$

As mentioned in Proposition 2, the total influence of  $\text{Maj}_n$  is  $\Theta(\sqrt{n})$ . Thus Boppana’s relation, Theorem 2, implies the following:

**Corollary 1.** *For any constant  $\epsilon < 1/2$ , every  $\epsilon$ -approximator for  $\text{Maj}_n$  with depth  $d$  requires size at least*

$$\exp\left(\Omega(n^{1/(2d-2)})\right).$$

*In particular, any  $\epsilon$ -approximating DNF for Majority requires size at least  $\exp(\Omega(\sqrt{n}))$ .*

This matches the upper bound we proved in Theorem 6, up to the constant in the exponent.

Due to space limitations, the proof of Theorem 7 is deferred to the full version of this paper. The main tool used therein is the Kruskal-Katona Theorem.

### 4 Falsifying the BKS Conjecture

In this section our goal is to falsify the BKS Conjecture. In particular, we will have to prove a lower bound for  $\epsilon$ -approximating a monotone function by DNF and CNF. Note that the technique we used in Section 3 — lower-bounding the total influence of an approximator and then using Theorem 2 — is useless here. This is because the BKS Conjecture was made as a converse to Theorem 2.

Since we have difficulty enough showing size lower bounds for  $\epsilon$ -approximating DNF, we should hope that our lower bounds for higher depths follow for an easy reason. This suggests looking for a counterexample among monotone functions with total influence  $\ll \log^2 n$ , since for such functions we will only have to prove sublinear size lower bounds for  $\epsilon$ -approximating circuits of depth  $d \geq 3$ .

The function we will use to falsify the BKS Conjecture will be based on the Tribes functions. These were originally introduced by Ben-Or and Linial [BOL90]; we will use slightly different parameters than they did, to simplify notation.

Given  $b \in \mathbb{N}$ , write  $I = \{1, 2, \dots, 2^b\}$ ,  $J = \{1, 2, \dots, b\}$ , and  $n = b2^b$ . We define the Tribes function  $\text{Tribes}_n : \{0, 1\}^n \rightarrow \{0, 1\}$  as follows. Given an input  $x \in \{0, 1\}^n$ , we index its bits as  $x_{i,j}$ , for  $i \in I$  and  $j \in J$ . We also write  $y_i = \bigwedge_{j \in J} x_{i,j}$ .  $\text{Tribes}_n(x)$  is then defined to be  $\bigvee_{i \in I} y_i$ .

In other words,  $\text{Tribes}_n$  is given by the monotone read-once DNF of width  $b$  and size  $2^b + 1$ . We have

$$\Pr_{\mathbf{x}}[\text{Tribes}_n(\mathbf{x}) = 1] = 1 - (1 - 2^{-b})^{2^b} \approx 1 - 1/e,$$

so  $\Pr_{\mathbf{x}}[\text{Tribes}_n(\mathbf{x}) = 1]$  is uniformly bounded away from 0 and 1.

We also define the monotone complement of  $\text{Tribes}_n$ :

$$\text{Tribes}_n^\dagger(x) = \overline{\text{Tribes}_n(\bar{x}_{1,1}, \bar{x}_{1,2}, \dots, \bar{x}_{2^b,b})}.$$

The function  $\text{Tribes}_n^\dagger(x)$  is given by the monotone read-once CNF of width  $b$  and size  $2^b + 1$ . It has  $\Pr[\text{Tribes}_n^\dagger(\mathbf{x}) = 1] \approx 1/e$ . Ben-Or and Linial showed that  $\mathbb{I}(\text{Tribes}_n) = \Theta(\log n)$ , and the same holds for  $\text{Tribes}_n^\dagger$  by boolean duality.

Suppose we attempt to approximate  $\text{Tribes}_n$  with some CNF  $C$ . We view  $C$  as being an AND of ORs, where each OR's wires may pass through a NOT gate before being wired to an input gate  $x_{i,j}$ .

Now further suppose we introduce additional “input gates”  $y_i$ , where each  $y_i$  is always equal to  $\bigwedge_{j \in J} x_{i,j}$ , and we allow the circuit  $C$  to access the  $y_i$  gates if it wants. Our main lemma uses the fact that  $\text{Tribes}_n$  depends only on the  $y_i$ 's to show that  $C$  can be taken to only depend on the  $y_i$ 's as well:

**Lemma 1.** *Suppose  $\text{Tribes}_n$  is  $\epsilon$ -approximated by a CNF  $C$  of size  $s$  and width  $w$  over the variables  $(x_{i,j})_{i \in I, j \in J}$ . Then there is another CNF  $C'$  of size at most  $s$  and width at most  $w$  only over the variables  $(y_i)_{i \in I}$  that also  $\epsilon$ -approximates  $\text{Tribes}_n$ .*

*Proof.* Given  $C$  over the input gates  $x_{i,j}$ , imagine that every wire going to an input gate  $x_{i,j}$  is instead rewired to access  $x_{i,j} \vee y_i$ . Call the resulting circuit  $C_1$ . We claim that  $C_1$  and  $C$  compute the same function of  $x$ . The reason is that on any input  $x$  where  $y_i = 0$ , the rewiring to  $x_{i,j} \vee y_i$  has no effect; and, on any input  $x$  where  $y_i = 1$ , the rewiring to  $x_{i,j} \vee y_i$  converts  $x_{i,j}$  to 1, but that still has no effect since  $y_i = 1 \Rightarrow x_{i,j} = 1$ . Since  $C$  was an  $\epsilon$ -approximator for  $\text{Tribes}_n$ , we have

$$\Pr_{\mathbf{x}}[C_1(\mathbf{x}, \mathbf{y}) \neq \text{Tribes}_n(\mathbf{x})] \leq \epsilon.$$

Now picking  $\mathbf{x}$  uniformly at random induces the  $2^{-b}$ -biased product distribution on  $\mathbf{y} \in \{0, 1\}^I$ . We can get the same distribution on  $(\mathbf{x}, \mathbf{y})$  by picking  $\mathbf{y}$  first and then picking  $\mathbf{x}$  conditioned on  $\mathbf{y}$ . I.e., for each  $i \in I$ : if  $y_i = 1$  then all  $x_{i,j}$ 's are chosen to be 1; if  $y_i = 0$  then the substring  $\mathbf{x}_i \in \{0, 1\}^J$  is chosen uniformly from  $\{0, 1\}^J \setminus \{(1, 1, \dots, 1)\}$ .

In view of this, and using the fact that  $\text{Tribes}_n$  depends only on  $\mathbf{y}$ , we have

$$\mathbb{E}_{\mathbf{y}} \left[ \Pr_{\mathbf{x} | \mathbf{y}} [C_1(\mathbf{x}, \mathbf{y}) \neq \text{Tribes}_n(\mathbf{y})] \right] \leq \epsilon.$$

We next introduce new input gates  $(z_{i,j})_{i \in I, j \in J}$  that take on random values, completely independent of the  $x_{i,j}$ 's and the  $y_i$ 's. Each substring  $(z_{i,j})_{j \in J}$  will

be uniform on  $\{0, 1\}^J \setminus \{(1, 1, \dots, 1)\}$ ; i.e., it will have the same distribution as  $(\mathbf{x}_i)_{i \in J} \mid \mathbf{y}_i = 0$ . Now let the circuit  $C_2$  be the same as  $C_1$  except with all accesses to the  $x_{i,j}$ 's replaced by accesses to the corresponding  $z_{i,j}$ 's.

We claim that for every string  $y \in \{0, 1\}^I$ , the distributions  $C_1(\mathbf{x} \mid y, y)$  and  $C_2(\mathbf{z}, y)$  are identical. The reason is that for each  $i \in I$  such that  $y_i = 1$ , the  $(\mathbf{x}_{i,j})_{j \in J}$  and  $(\mathbf{z}_{i,j})_{j \in J}$  values are irrelevant, since  $C_1$  only accesses  $x_{i,j}$  by accessing  $x_{i,j} \vee y_i$  and the same is true of  $C_2$  and  $z_{i,j}$ . On the other hand, for each  $i \in I$  such that  $y_i = 0$ , the  $(\mathbf{x}_{i,j})_{j \in J}$  and  $(\mathbf{z}_{i,j})_{j \in J}$  values are identically distributed.

In light of this, we conclude

$$\mathbf{E}_{\mathbf{y}} \left[ \Pr_{\mathbf{z}} \left[ C_2(\mathbf{z}, \mathbf{y}) \neq \text{Tribes}_n(\mathbf{y}) \right] \right] \leq \epsilon,$$

which can be switched to

$$\mathbf{E}_{\mathbf{z}} \left[ \Pr_{\mathbf{y}} \left[ C_2(\mathbf{z}, \mathbf{y}) \neq \text{Tribes}_n(\mathbf{y}) \right] \right] \leq \epsilon.$$

Since  $\mathbf{z}$  and  $\mathbf{y}$  are independent, we can conclude there must be a particular setting  $z^*$  such that

$$\Pr_{\mathbf{y}} \left[ C_2(z^*, \mathbf{y}) \neq \text{Tribes}_n(\mathbf{y}) \right] \leq \epsilon.$$

We may now take  $C'$  to be the circuit only over the  $\mathbf{y}$  gates gotten by fixing the input  $z^*$  for  $C_2$ . It is easy to check that  $C'$  has width at most  $w$  and size at most  $s$ . □

We can now use Lemma 11 to show that  $\text{Tribes}_n$  has no good CNF approximator of width much smaller than  $n/\log n$ :

**Theorem 8.** *Any CNF that .2-approximates  $\text{Tribes}_n$  must have width at least  $(1/3)2^b = \Omega(n/\log n)$ .*

*Proof.* Let  $C$  be a CNF of width  $w$  that .2-approximates  $\text{Tribes}_n$  over the variables  $(x_{i,j})_{i \in I, j \in J}$ . Using Lemma 11, convert it to a CNF  $C'$  over the variables  $(y_i)_{i \in I}$  that .2-approximates  $\text{Tribes}_n$ . We may assume that no term in  $C'$  includes both  $y_i$  and  $\bar{y}_i$  for some  $i$ . We now consider two cases.

Case 1: Every term in  $C'$  includes at least one negated  $y_i$ . In this case,  $C'$  is 1 whenever  $y = (0, 0, \dots, 0)$ . But  $\text{Tribes}_n$  is 0 when  $y = (0, 0, \dots, 0)$ . Since this occurs with probability  $(1 - 2^{-b})^{2^b} \geq 1/4 > .2$ , we have a contradiction.

Case 2:  $C'$  has at least one term in which all  $y_i$ 's are unnegated. Suppose this term has width  $w$ . Since  $y_i$  is true only with probability  $2^{-b}$ , this term is true with probability at most  $w2^{-b}$ , by the union bound. And whenever this term is false,  $C'$  is false. Hence  $\Pr[C' = 0] \geq 1 - w2^{-b}$ . Since  $\Pr[\text{Tribes}_n = 0] \leq 1/e$  and  $C'$  is a .2-approximator for  $\text{Tribes}_n$ , we must have  $1 - w2^{-b} \leq 1/e + .2 \Rightarrow w2^{-b} \geq 1/3$ , completing the proof. □

By symmetry of 0 and 1, we infer:

**Corollary 2.** *Any DNF that .2-approximates  $\text{Tribes}_n^\dagger$  must have width at least  $\Omega(n/\log n)$ .*

As an aside, we can now show that the idea of approximating TOPs by DNFs discussed in Section 1.3 cannot work. Since  $\text{Tribes}_n^\dagger$  is computable by a polynomial-size CNF, Jackson's Harmonic Sieve learning algorithm [Jac95] can produce a polynomial-size  $O(\log n)$ -width TOP  $\epsilon$ -approximator for it, for any constant  $\epsilon > 0$ . But one can never convert this to even a .2-approximating DNF of size smaller than  $2^{\Omega(n/\log n)}$ , by Corollary 2 combined with Observation 1.

We now define the function that contradicts the BKS Conjecture:

**Definition 3.** *Let  $n$  be of the form  $b2^{b+1}$ . We define  $\mathcal{F}_n : \{0, 1\}^n \rightarrow \{0, 1\}$  to be the OR of  $\text{Tribes}_{n/2}$  and  $\text{Tribes}_{n/2}^\dagger$ , on disjoint sets of bits.*

**Proposition 3.**  *$\mathcal{F}_n$  is a monotone function computable by a depth-3 read-once formula, and  $\mathbb{I}(\mathcal{F}) = O(\log n)$ .*

The fact that  $\mathbb{I}(\mathcal{F}_n) = O(\log n)$  holds because  $\mathbb{I}(\mathcal{F}_n) \leq \mathbb{I}(\text{Tribes}_{n/2}) + \mathbb{I}(\text{Tribes}_{n/2}^\dagger) = O(\log n) + O(\log n)$ .

**Theorem 9.** *Any depth-2 circuit that .04-approximates  $\mathcal{F}_n$  must have size at least  $2^{\Omega(n/\log n)}$ .*

*Proof.* Suppose  $D$  is a DNF of size  $s$  that .04-approximates  $\mathcal{F}_n$ . By Observation 1, we can replace it with a DNF  $D'$  of width at most  $\log(100s)$  which  $.04 + 1/100 = .05$ -approximates  $\mathcal{F}_n$ .

Consider choosing  $\mathbf{x} \in \{0, 1\}^{n/2}$  uniformly at random from the set of strings that make  $\text{Tribes}_{n/2}$  false, and also choosing  $\mathbf{y} \in \{0, 1\}^{n/2}$  independently and uniformly at random. Since at least  $1/4$  of all strings make  $\text{Tribes}_{n/2}$  false (close to  $1/e$ , in fact), this distribution is uniform on some subset of  $\{0, 1\}^n$  of fractional size at least  $1/4$ . Since  $D'$  errs in computing  $\mathcal{F}_n$  on at most a .05 fraction of strings, we conclude that

$$\Pr[D'(\mathbf{x}, \mathbf{y}) \neq \mathcal{F}_n(\mathbf{x}, \mathbf{y})] \leq 4 \cdot .05 = .2.$$

Note that  $\mathcal{F}_n(\mathbf{x}, \mathbf{y})$  is always just  $\text{Tribes}_{n/2}^\dagger(\mathbf{y})$ . We conclude that there must be a particular setting of bits  $x^* \in \{0, 1\}^{n/2}$  such that

$$\Pr[D'(x^*, \mathbf{y}) \neq \text{Tribes}_{n/2}^\dagger(\mathbf{y})] \leq .2.$$

Hence we have a DNF  $D'' = D'(x^*, \cdot)$  over  $\{0, 1\}^{n/2}$  of width at most  $\log(100s)$  that .2-approximates  $\text{Tribes}_{n/2}^\dagger$ . By Corollary 2, we conclude that  $\log(100s) \geq \Omega(n/\log n)$ . Hence the original DNF  $D$  has size at least  $2^{\Omega(n/\log n)}$ .

A very similar argument, restricting to the inputs to  $\mathcal{F}_n$  where the  $\text{Tribes}_{n/2}^\dagger$  part is 0 and then using Theorem 8 shows that any CNF that is a .04-approximator for  $\mathcal{F}_n$  must have size at least  $2^{\Omega(n/\log n)}$ . This completes the proof.  $\square$

Theorem 9 already implies that the BKS Conjecture cannot hold with  $d$  always equal to 2. To completely falsify the conjecture, we need the following additional observations:

**Proposition 4.** *Any function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  that .02-approximates  $\mathcal{F}_n$  must depend on at least  $\Omega(n)$  input bits.*

*Proof.* It is very well known (see [DF06] for a written proof) that there is an explicit  $\epsilon > 0$  (and certainly  $\epsilon = .1$  is achievable) such that any function  $g : \{0, 1\}^n \rightarrow \{0, 1\}$  that  $\epsilon$ -approximates  $\text{Tribes}_{n/2}$  must depend on at least  $\Omega(n)$  of its input bits. Now an argument very similar to the one used in the proof of Theorem 9 shows that if  $f$  is a .02-approximator for  $\mathcal{F}_n$ , then some restriction of  $f$  must be a  $\delta$ -approximator for  $\text{Tribes}_{n/2}$  with  $\delta \leq 4 \cdot .02 < .1$ . Since this restriction must depend on at least  $\Omega(n/2)$  input bits, we conclude that  $f$  must also depend on at least this many input bits.  $\square$

**Proposition 5.** *Any circuit that .01-approximates  $\mathcal{F}_n$  must have size at least  $\Omega(n/\log n)$ .*

*Proof.* Suppose the circuit  $C$  has size  $s$  and is a .01-approximator for  $\mathcal{F}_n$ . By Observation 1, there is another circuit  $C'$  of size at most  $s$  and width at most  $\log(100s)$  that .01-approximates  $C$ ; this  $C'$  is thus a  $.01 + .01 = .02$ -approximator for  $\mathcal{F}_n$ . But  $C'$  depends on at most size  $\times$  width  $= s \log(100s)$  literals. Hence  $s \log(100s) \geq \Omega(n)$  by Proposition 4 and so  $s \geq \Omega(n/\log n)$ .  $\square$

Finally, we've established:

**Theorem 10.** *The BKS Conjecture is false.*

*Proof.* We use the function  $\mathcal{F}_n$ , which is monotone and has  $\mathbb{I}(\mathcal{F}_n) = O(\log n)$ . The BKS Conjecture implies that there is some universal constant  $K = K(.01) < \infty$  such that the following holds: There is a circuit  $C$  that .01-approximates  $\mathcal{F}_n$  and has depth  $d$  and size  $s$ , for some  $d$  and  $s$  satisfying

$$s \leq \exp\left(\left(K \cdot \mathbb{I}(\mathcal{F}_n)\right)^{1/(d-1)}\right) = \exp\left(O(\log^{1/(d-1)} n)\right).$$

Now  $d$  can't be 2, since this would imply  $s \leq \text{poly}(n)$ , and we know from Theorem 9 that there is no circuit .01-approximating  $\mathcal{F}_n$  of depth 2 and size  $2^{o(n/\log n)}$ . But  $d \geq 3$  is also impossible, since this would imply  $s \leq \exp(\sqrt{\log n})$ , and we know from Proposition 5 that there is no circuit .01-approximating  $\mathcal{F}_n$  of size  $o(n/\log n)$ .  $\square$

## References

- [Ajt83] Ajtai, M.:  $\Sigma_1^1$ -formulae on finite structures. *Annals of Pure and Applied Logic* 24, 1–48 (1983)
- [Ajt93] Ajtai, M.: Approximate counting with uniform constant-depth circuits. In: *Advances in Computational Complexity Theory*, pp. 1–20. Amer. Math. Soc., Providence, RI (1993)

- [AKS83] Ajtai, M., Komlós, J., Szemerédi, E.: Sorting in  $c \log n$  parallel steps. *Combinatorica* 3, 1–19 (1983)
- [BKS99] Benjamini, I., Kalai, G., Schramm, O.: Noise sensitivity of boolean functions and applications to percolation. *Inst. Hautes Études Sci. Publ. Math.* 90, 5–43 (1999)
- [BOL90] Ben-Or, M., Linial, N.: Collective coin flipping. In: Micali, S. (ed.) *Randomness and Computation*, Academic Press, New York (1990)
- [Bop86] Boppana, R.: Threshold functions and bounded depth monotone circuits. *J. Comp. Sys. Sci.* 32(2), 222–229 (1986)
- [Bop97] Boppana, R.: The average sensitivity of bounded-depth circuits. *Inf. Process. Lett.* 63(5), 257–261 (1997)
- [BT96] Bshouty, N., Tamon, C.: On the Fourier spectrum of monotone functions. *Journal of the ACM* 43(4), 747–770 (1996)
- [DF06] Dinur, I., Friedgut, E.: Lecture notes (2006), available at <http://www.cs.huji.ac.il/~analyt/scribes/L11.pdf>
- [Fri98] Friedgut, E.: Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica* 18(1), 474–483 (1998)
- [Fri99] Friedgut, E.: Sharp thresholds of graph properties, and the  $k$ -SAT problem. *J. American Math. Soc.* 12(4), 1017–1054 (1999)
- [Fri05] Friedgut, E.: Hunting for sharp thresholds. *Random Struct. & Algorithms* 26(1-2), 37–51 (2005)
- [Hås86] Håstad, J.: *Computational Limitations for Small Depth Circuits*. MIT Press, Cambridge, MA (1986)
- [HMP06] Hoory, S., Magen, A., Pitassi, T.: Monotone circuits for the majority function. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) *APPROX 2006 and RANDOM 2006*. LNCS, vol. 4110, Springer, Heidelberg (2006)
- [Jac95] Jackson, J.: The Harmonic sieve: a novel application of Fourier analysis to machine learning theory and practice. PhD thesis, Carnegie Mellon University, (August 1995)
- [Kal00] Kalai, G.: *Combinatorics with a geometric flavor: some examples*, 2000. GFAFA Special Volume 10, Birkhauser Verlag, Basel (2000)
- [KKL88] Kahn, J., Kalai, G., Linial, N.: The influence of variables on boolean functions. In: *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pp. 68–80 (1988)
- [KS05] Kalai, G., Safra, S.: Threshold phenomena and influence. In: *Computational Complexity and Statistical Physics*, Oxford University Press, Oxford (2005)
- [LMN93] Linial, N., Mansour, Y., Nisan, N.: Constant depth circuits, Fourier transform and learnability. *Journal of the ACM* 40(3), 607–620 (1993)
- [Mar74] Margulis, G.: Probabilistic characteristics of graphs with large connectivity. *Prob. Peredachi Inform.* 10, 101–108 (1974)
- [OS06] O'Donnell, R., Servedio, R.: Learning monotone decision trees in polynomial time. *SIAM J. Comp.* (to appear, 2006)
- [PPZ92] Paterson, M., Pippenger, N., Zwick, U.: Optimal carry save networks. *Boolean function complexity* 169, 174–201 (1992)
- [Rus78] Russo, L.: On the critical percolation probabilities. *Z. Wahrsch. Werw. Gebiete* 43, 39–48 (1978)
- [Tal96] Talagrand, M.: How much are increasing sets positively correlated? *Combinatorica* 16(2), 243–258 (1996)
- [Val84] Valiant, L.: Short monotone formulae for the majority function. *J. Algorithms* 5(3), 363–366 (1984)
- [Vio05] Viola, E.: On probabilistic time versus alternating time. *ECCC 2005*, 173 (2005)

# Exotic Quantifiers, Complexity Classes, and Complete Problems

## (Extended Abstract)

Peter Bürgisser<sup>1,\*</sup> and Felipe Cucker<sup>2,\*\*</sup>

<sup>1</sup> Dept. of Mathematics, University of Paderborn, D-33095 Paderborn, Germany  
pbuerg@upb.de

<sup>2</sup> Dept. of Mathematics, City University of Hong Kong, Hong Kong  
macucker@math.cityu.edu.hk

**Abstract.** We define new complexity classes in the Blum-Shub-Smale theory of computation over the reals, in the spirit of the polynomial hierarchy, with the help of infinitesimal and generic quantifiers. Basic topological properties of semialgebraic sets like boundedness, closedness, compactness, as well as the continuity of semialgebraic functions are shown to be complete in these new classes. All attempts to classify the complexity of these problems in terms of the previously studied complexity classes have failed. We also obtain completeness results in the Turing model for the corresponding discrete problems. In this setting, it turns out that infinitesimal and generic quantifiers can be eliminated, so that the relevant complexity classes can be described in terms of usual quantifiers only.

## 1 Introduction

The complexity theory over the real numbers introduced by L. Blum, M. Shub, and S. Smale developed quickly after their foundational paper [4]. Complexity classes other than  $P_{\mathbb{R}}$  and  $NP_{\mathbb{R}}$  were introduced (e.g., in [8,17,11]), completeness results were proven (e.g., in [8,17,22]), separations were obtained ([10,16]), machine-independent characterizations of complexity classes were exhibited ([6,14,18]).

There are two points in this development which we would like to stress. Firstly, all the considered complexity classes were natural versions over the real numbers of existing complexity classes in the classical setting. Secondly, the catalogue of completeness results is disappointingly small. For a given semialgebraic set  $S \subseteq \mathbb{R}^n$ , deciding whether a point in  $\mathbb{R}^n$  belongs to  $S$  is  $P_{\mathbb{R}}$ -complete [17], deciding whether  $S$  is non-empty (or non-convex, or of dimension at least  $d$  for a given  $d \in \mathbb{N}$ ) is  $NP_{\mathbb{R}}$ -complete [4,15,22], and computing its Euler-Yao characteristic is  $FP_{\mathbb{R}}^{\#P_{\mathbb{R}}}$ -complete [8]. That is, essentially, all.

---

\* Partially supported by DFG grant BU 1371 and Paderborn Institute for Scientific Computation.

\*\* Partially supported by City University SRG grant 7001712.



Yet, there are plenty of natural problems involving semialgebraic sets: computing local dimensions, deciding denseness, closedness, unboundedness, etc. Consider, for instance, the latter. We can express that  $S$  is unbounded by

$$\forall K \in \mathbb{R} \exists x \in \mathbb{R}^n (x \in S \wedge \|x\| \geq K). \tag{1}$$

Properties describable with expressions like this one are common in classical complexity theory and in recursive function theory. Extending an idea by Kleene [19] for the latter, Stockmeyer introduced in [24] the polynomial time hierarchy which is built on top of NP and coNP in a natural way.<sup>1</sup> Recall, a set  $S$  is in NP when there is a polynomial time decidable relation  $R$  such that, for every  $x \in \{0, 1\}^*$ ,

$$x \in S \iff \exists y \in \{0, 1\}^{\text{size}(x)^{O(1)}} R(x, y).$$

The class coNP is defined replacing  $\exists$  by  $\forall$ . Classes in the polynomial hierarchy are then defined by allowing the quantifiers  $\exists$  and  $\forall$  to alternate (with a bounded number of alternations). If there are  $k$  alternations of quantifiers, we obtain the classes  $\Sigma^{k+1}$  (if the first quantifier is  $\exists$ ) and  $\Pi^{k+1}$  (if the first quantifier is  $\forall$ ). Note that  $\Sigma^1 = \text{NP}$  and  $\Pi^1 = \text{coNP}$ . The definition of these classes over  $\mathbb{R}$  is straightforward [3, Ch. 21].

It follows thus from (1) that deciding unboundedness is in  $\Pi_{\mathbb{R}}^2$ , the universal second level of the polynomial hierarchy over  $\mathbb{R}$ . On the other hand, it is easy to prove that this problem is NP $_{\mathbb{R}}$ -hard. But we do not have completeness for any of these two classes.

A similar situation appears for deciding denseness. We can express that  $S \subseteq \mathbb{R}^n$  is Euclidean dense by

$$\forall x \in \mathbb{R}^n \forall \varepsilon > 0 \exists y \in \mathbb{R}^n (y \in S \wedge \|x - y\| \leq \varepsilon)$$

thus showing that this problem is in  $\Pi_{\mathbb{R}}^2$ . But we can not prove hardness in this class. Actually, we can not even manage to prove NP $_{\mathbb{R}}$ -hardness or coNP $_{\mathbb{R}}$ -hardness. Yet a similar situation occurs with closedness, which is in  $\Pi_{\mathbb{R}}^3$  since we express that  $S$  is closed by

$$\forall x \in \mathbb{R}^n \exists \varepsilon > 0 \forall y \in \mathbb{R}^n (x \notin S \wedge \|x - y\| \leq \varepsilon \Rightarrow y \notin S)$$

but the best hardness result we can prove is coNP $_{\mathbb{R}}$ -hardness. It would seem that the landscape of complexity classes between P $_{\mathbb{R}}$  and the third level of the polynomial hierarchy is not enough to capture the complexity of the problems above.

A main goal of this paper is to show that the two features we pointed out earlier namely, a theory uniquely based upon real versions of classical complexity classes, and a certain scarcity of completeness results, are not unrelated. With the help of infinitesimal and generic quantifiers we shall define complexity classes

---

<sup>1</sup> All along this paper we use a subscript  $\mathbb{R}$  to differentiate complexity classes over  $\mathbb{R}$  from discrete complexity classes. To further emphasize this difference, we use **sans serif** to denote the latter.

lying in between the different levels of the polynomial hierarchy. These new classes will allow us to determine the complexity of some of the problems we mentioned (and of others we didn't mention) or, in some cases, to decrease the gap between their lower and upper complexity bounds as we know them today.

A remarkable feature of these classes is that, as with the classes in the polynomial hierarchy, they are defined using quantifiers which act as operators on complexity classes. The properties of these operators naturally become an object of study for us. Thus, another goal of this paper is to provide some structural results for these operators.

We remark that a similar classification has already been achieved in the so called additive BSS model, without the need to introduce exotic quantifiers [7,9].

## 2 Preliminaries

We assume some basic knowledge on real machines and complexity as presented, for instance, in [3,4].

An *algebraic circuit*  $\mathcal{C}$  over  $\mathbb{R}$  is an acyclic directed graph where each node has indegree 0, 1 or 2. Nodes with indegree 0 are either labeled as *input nodes* or with elements of  $\mathbb{R}$  (we shall call them *constant nodes*). Nodes with indegree 2 are labeled with the binary operators of  $\mathbb{R}$ , i.e., one of  $\{+, \times, -, /\}$ . They are called *arithmetic nodes*. Nodes with indegree 1 are either *sign nodes* or *output nodes*. All the output nodes have outdegree 0. Otherwise, there is no upper bound for the outdegree of the other kinds of nodes. For an algebraic circuit  $\mathcal{C}$ , the *size* of  $\mathcal{C}$ , is the number of nodes in  $\mathcal{C}$ . The *depth* of  $\mathcal{C}$ , is the length of the longest path from some input node to some output node.

An arithmetic node computes a function of its input values in an obvious manner. Sign nodes compute the function  $\text{sgn}$  defined by  $\text{sgn}(x) = 1$  if  $x \geq 0$  and  $\text{sgn}(x) = 0$  otherwise. To a circuit  $\mathcal{C}$  with  $n$  input gates and  $m$  output gates is associated a function  $f_{\mathcal{C}} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ . This function may not be total since divisions by zero may occur (in which case, by convention,  $f_{\mathcal{C}}$  is not defined on its input). We say that an algebraic circuit is a *decision* circuit if it has only one output gate whose parent is a sign gate. Thus, a decision circuit  $\mathcal{C}$  with  $n$  input gates computes a function  $f_{\mathcal{C}} : \mathbb{R}^n \rightarrow \{0, 1\}$ . The set *decided* by the circuit is

$$S_{\mathcal{C}} = \{x \in \mathbb{R}^n \mid f_{\mathcal{C}}(x) = 1\}.$$

Subsets of  $\mathbb{R}^n$  decidable by algebraic circuits are known as *semialgebraic sets*. They are defined as those sets which can be written as a Boolean combination of solution sets of polynomial inequalities  $\{x \in \mathbb{R}^n \mid f(x) \geq 0\}$ .

Semialgebraic sets will be inputs to problems considered in this paper. They will be either given by a Boolean combination of polynomial equalities and inequalities or by a decision circuit. If not otherwise specified, we mean the first variant. In this case, polynomials are encoded with the so called *dense encoding*, i.e., they are represented by the complete list of their coefficients (including zero coefficients).

We close this section by recalling a completeness result, which will play an important role in our developments. For  $d \in \mathbb{N}$  let  $\text{DIM}_{\mathbb{R}}(d)$  be the problem of, given a semialgebraic set  $S$ , deciding whether  $\dim S \geq d$ . In [22] Koiran proved that  $\text{DIM}_{\mathbb{R}}$  is  $\text{NP}_{\mathbb{R}}$ -complete.

### 2.1 Infinitesimal and Generic Quantifiers

We are going to define three logical quantifiers in the theory of the reals. Suppose  $\varphi(\varepsilon)$  is a formula with one free variable  $\varepsilon$ . The expression  $\text{H}\varepsilon\varphi(\varepsilon)$  shall express that  $\varphi(\varepsilon)$  holds for *sufficiently small real*  $\varepsilon > 0$ , that is,

$$\text{H}\varepsilon\varphi(\varepsilon) \stackrel{\text{def}}{\equiv} \exists\mu > 0 \forall\varepsilon \in (0, \mu) \varphi(\varepsilon). \tag{2}$$

Suppose that  $\psi(x)$  is a formula with  $n$  free variables  $x_1, \dots, x_n$ . We shall write  $\forall^*x\psi(x)$  in order to express that *almost all*  $x \in \mathbb{R}^n$  (with respect to the *Euclidean topology*) satisfy  $\psi(x)$ . Explicitly,

$$\forall^*x\psi(x) \stackrel{\text{def}}{\equiv} \forall x_0 \forall\varepsilon > 0 \exists x (\|x - x_0\| < \varepsilon \wedge \psi(x)). \tag{3}$$

If we put  $S_\psi = \{x \in \mathbb{R}^n \mid \psi(x) \text{ holds}\}$  this is equivalent to  $\dim(\mathbb{R}^n - S_\psi) < n$ , as  $S_\psi$  is semialgebraic, cf. [5]. Furthermore, we shall write  $\exists^*x\psi(x)$  to express that *almost all*  $x \in \mathbb{R}^n$  (with respect to the *Zariski topology*) satisfy  $\psi(x)$ . This is the case iff  $\dim S_\psi = n$ , which is in turn equivalent to

$$\exists^*x\psi(x) \stackrel{\text{def}}{\equiv} \exists x_0 \exists\varepsilon > 0 \forall x (\|x - x_0\| < \varepsilon \Rightarrow \psi(x)), \tag{4}$$

which expresses that  $S_\psi$  contains an open ball. (For a proof of this equivalence see [5].) The generic quantifiers  $\forall^*$  and  $\exists^*$  were previously introduced by Koiran [22], while the infinitesimal quantifier  $\text{H}$  so far hasn't been studied in a complexity framework.

By definition,  $\exists^*\psi(x)$  is equivalent to  $\neg(\forall^*\neg\psi(x))$ . By contrast, it is easy to see that the quantifier  $\text{H}$  allows to pull in negations:  $\neg\text{H}\varepsilon\varphi(\varepsilon)$  is equivalent to  $\text{H}\varepsilon\neg\varphi(\varepsilon)$ .

We are next going to interpret the new quantifiers as operators acting on complexity classes. We denote by  $\mathbb{R}^\infty$  the disjoint union  $\sqcup_{n \geq 0} \mathbb{R}^n$ . If  $x \in \mathbb{R}^n \subset \mathbb{R}^\infty$  we define its *size* to be  $|x| = n$ .

**Definition 2.1.** *Let  $\mathcal{C}$  be a complexity class of decision problems.*

1. *The class  $\text{HC}$  consists of the  $A \subseteq \mathbb{R}^\infty$  such that there exists  $B \subseteq \mathbb{R} \times \mathbb{R}^\infty$ ,  $B \in \mathcal{C}$ , such that, for all  $x \in \mathbb{R}^\infty$ ,*

$$x \in A \iff \text{H}\varepsilon(\varepsilon, x) \in B.$$

2. *Let  $Q$  be one of the quantifiers  $\forall, \forall^*, \exists, \exists^*$ . The class  $QC$  consists of the  $A \subseteq \mathbb{R}^\infty$  such that there exists a polynomial  $p$  and  $B \subseteq \mathbb{R}^\infty \times \mathbb{R}^\infty$ ,  $B \in \mathcal{C}$ , such that, for all  $x \in \mathbb{R}^\infty$ ,*

$$x \in A \iff Qz \in \mathbb{R}^{p(|x|)}(z, x) \in B.$$

By repeatedly applying these operators to  $P_{\mathbb{R}}$  we may define many new complexity classes, which can be seen as a refinement of the polynomial hierarchy over the reals. These classes somehow take into account the *topology* of  $\mathbb{R}$ , an aspect completely absent in the discrete setting.

In order to simplify notation we will omit  $P_{\mathbb{R}}$  and write simply  $NP_{\mathbb{R}} = \exists P_{\mathbb{R}} = \exists$ ,  $coNP_{\mathbb{R}} = \forall P_{\mathbb{R}} = \forall$  etc. We call the classes defined this way *polynomial classes*. It is easy to see that they are closed under many-one reductions. Completeness shall always refer to such reductions.

### 2.2 Standard Complete Problems

Let  $STANDARD(H\exists)$  be the problem of deciding, given a polynomial  $f$  in  $n + 1$  variables (in dense encoding), whether

$$H\epsilon \exists x \in \mathbb{R}^n f(\epsilon, x) = 0.$$

The problem  $STANDARD(H\forall)$  is analogously defined by requiring  $f(\epsilon, x) \neq 0$  instead. The usual proof of  $NP_{\mathbb{R}}$ -completeness of the real feasibility problem [3,4] yields:

**Proposition 2.2.**  *$STANDARD(H\exists)$  is  $H\exists$ -complete and  $STANDARD(H\forall)$  is  $H\forall$ -complete.*

We remark that any polynomial class can be shown to have a standard complete problem.

### 3 Natural Problems Complete for $H\exists$ and $H\forall$

Consider the following problems

- $UNBOUNDED_{\mathbb{R}}$  (*Unboundedness*) Given a semialgebraic set  $S$ , is it unbounded?
- $EADH_{\mathbb{R}}$  (*Euclidean Adherence*) Given a semialgebraic set  $S$  and a point  $x$ , decide whether  $x$  belongs to the Euclidean closure  $\overline{S}$  of  $S$ .
- $LOCDIM_{\mathbb{R}}$  (*Local Dimension*) Given a semialgebraic set  $S \subseteq \mathbb{R}^n$ , a point  $x \in S$ , and  $d \in \mathbb{N}$ , is  $\dim_x S \geq d$ ?

**Proposition 3.1.**  *$UNBOUNDED_{\mathbb{R}}$ ,  $EADH_{\mathbb{R}}$ , and  $LOCDIM_{\mathbb{R}}$  are  $H\exists$ -complete.*

PROOF. A set  $S$  is unbounded if and only if

$$H\epsilon \exists x \in \mathbb{R}^n (\epsilon \|x\| \geq 1 \wedge x \in S).$$

This shows  $UNBOUNDED_{\mathbb{R}} \in H\exists$ . In a similar way one sees that  $EADH_{\mathbb{R}} \in H\exists$ .

Let  $B(x, \epsilon)$  denote the open  $\epsilon$ -ball centered at  $x$ . From the equivalence

$$\dim_x S \geq d \iff H\epsilon \dim(S \cap B(x, \epsilon)) \geq d$$

and the fact [22] that  $DIM_{\mathbb{R}} \in NP_{\mathbb{R}}$  we conclude  $LOCDIM_{\mathbb{R}} \in H\exists$ .

For showing hardness, consider the auxiliary problem  $\mathcal{L} \subseteq \mathbb{R}^\infty$  consisting of, given  $g \in \mathbb{R}[\varepsilon, X_1, \dots, X_n]$ , deciding whether

$$\text{H}\varepsilon \exists t \in (-1, 1)^n \ g(\varepsilon, t_1, \dots, t_n) = 0.$$

We first reduce  $\text{STANDARD}(\text{H}\exists)$  to  $\mathcal{L}$ , which will show that  $\mathcal{L}$  is  $\text{H}\exists$ -complete, cf. Proposition 2.2. To do so, note that the existence of a root in  $\mathbb{R}^n$  of a polynomial  $f$  is equivalent to the existence of a root in the open unit cube  $(-1, 1)^n$  for a suitable other polynomial. This is so since the mapping  $\psi(\lambda) = \frac{\lambda}{1-\lambda^2}$  bijects  $(-1, 1)$  with  $\mathbb{R}$ . Therefore, for  $f \in \mathbb{R}[Y, X_1, \dots, X_n]$ ,

$$\text{H}\varepsilon \exists x \in \mathbb{R}^n \ f(\varepsilon, x_1, \dots, x_n) = 0 \iff \text{H}\varepsilon \exists t \in (-1, 1)^n \ g(\varepsilon, t_1, \dots, t_n) = 0,$$

where  $d_i = \deg_{x_i} f$  and  $g \in \mathbb{R}[Y, T_1, \dots, T_n]$  is given by

$$g(\varepsilon, t_1, \dots, t_n) := (1 - t_1^2)^{d_1} (1 - t_2^2)^{d_2} \dots (1 - t_n^2)^{d_n} f(\varepsilon, \psi(t_1), \dots, \psi(t_n)).$$

Note that we can construct  $g$  in time polynomial in the size of  $f$ . (As we are representing  $f$  and  $g$  in the dense encoding, the divisions can be eliminated in polynomial time.) So the mapping  $f \mapsto g$  indeed reduces  $\text{STANDARD}(\text{H}\exists)$  to  $\mathcal{L}$ .

In order to reduce  $\mathcal{L}$  to  $\text{UNBOUNDED}_{\mathbb{R}}$  we associate to  $g \in \mathbb{R}[Y, T_1, \dots, T_n]$  the semialgebraic set  $S := \{(y, t) \in \mathbb{R} \times (-1, 1)^n \mid h(y, t) = 0\}$ , where  $h$  is the polynomial defined by  $h(Y, T) = Y^{2 \deg_Y g} g(1/Y^2, T)$ . Then  $g \in \mathcal{L}$  if and only if  $S$  is unbounded. This proves that  $\text{UNBOUNDED}_{\mathbb{R}}$  is  $\text{H}\exists$ -complete.

We reduce now  $\text{UNBOUNDED}_{\mathbb{R}}$  to  $\text{EADH}_{\mathbb{R}}$ . To a polynomial  $f$  of degree  $d$  in  $n$  variables we assign  $f' := \|X\|^{2d} f(\|X\|^{-2} X)$ . Let  $S \subseteq \mathbb{R}^n$  be a semialgebraic set given by a Boolean combination of inequalities of the form  $f(x) > 0$ . Without loss of generality,  $0 \notin S$ . The set defined by the same Boolean combination of the inequalities  $f'(x) > 0$  and the condition  $x \neq 0$  is the image of  $S$  under the inversion map  $i: \mathbb{R}^n \setminus \{0\} \rightarrow \mathbb{R}^n \setminus \{0\}, x \mapsto \|x\|^{-2} x$ . Hence  $S$  is unbounded if and only if  $0$  belongs to the closure of  $i(S) \setminus \{0\}$ .

Finally, it is easy to reduce  $\text{EADH}_{\mathbb{R}}$  to  $\text{LOCDIM}_{\mathbb{R}}$ . For given  $S \subseteq \mathbb{R}^n$  and  $x \in \mathbb{R}^n$  put take  $S' = \mathbb{R}^n$  if  $x \in S$ . Else, put  $S' = S \cup \{x\}$ . Then  $x \in \overline{S}$  iff  $\dim_x S' \geq 1$ . □

A *basic semialgebraic set* is the solution set  $S \subseteq \mathbb{R}^n$  of a system of polynomial equalities and inequalities of the form

$$f = 0, h_1 \geq 0, \dots, h_p \geq 0, g_1 > 0, \dots, g_q > 0. \tag{5}$$

Consider the following problems:

$\text{BASICCLOSED}_{\mathbb{R}}$  (*Closedness for basic semialgebraic sets*) Given a basic semialgebraic set  $S$ , is it closed?

$\text{BASICCOMPACT}_{\mathbb{R}}$  (*Compactness for basic semialgebraic sets*) Given a basic semialgebraic set  $S$ , is it compact?

**Theorem 3.2.**  $\text{BASICCLOSED}_{\mathbb{R}}$  and  $\text{BASICCOMPACT}_{\mathbb{R}}$  are  $\text{H}\forall$ -complete.

The proof needs some preparation. For a basic semialgebraic set  $S$  given as in (5) define for  $\varepsilon > 0$

$$S_\varepsilon = \{f = 0, h_1 \geq 0, \dots, h_p \geq 0, g_1 \geq \varepsilon, \dots, g_q \geq \varepsilon\}.$$

Note that  $S_\varepsilon \subseteq S_{\varepsilon'} \subseteq S$  for  $0 < \varepsilon' < \varepsilon$  and that  $S = \bigcup_{\varepsilon > 0} S_\varepsilon$ .

**Lemma 3.3.** *Suppose that  $K^S := \{f = 0, h_1 \geq 0, \dots, h_p \geq 0\}$  is bounded. Then  $S$  is closed iff  $S_\varepsilon = S$  for sufficiently small  $\varepsilon > 0$ .*

The condition  $H\varepsilon(S_\varepsilon = S)$  is testable in  $H\forall$ . For showing membership of  $\text{BASICCLOSED}_\mathbb{R}$  to  $H\forall$  it is therefore sufficient to reduce the general situation to the one with bounded  $K^S$ .

Let  $S^n$  denote the  $n$ -dimensional unit sphere and  $\mathcal{N} = (0, \dots, 0, 1)$ . The *stereographic projection*  $\pi : S^n - \{\mathcal{N}\} \rightarrow \mathbb{R}^n, (x, t) \mapsto \frac{1}{1-t}x$  is a homeomorphism. Consider  $\tilde{S} := \pi^{-1}(S) \cup \{\mathcal{N}\}$ . Then,  $\tilde{S}$  is a basic semialgebraic set such that  $K^{\tilde{S}}$  is bounded. Moreover,  $S$  is closed in  $\mathbb{R}^n$  iff  $\tilde{S}$  is closed in  $\mathbb{R}^{n+1}$ . This shows membership of  $\text{BASICCLOSED}_\mathbb{R}$  to  $H\forall$ . The claimed membership of  $\text{BASICCOMPACT}_\mathbb{R}$  follows now by using  $\text{UNBOUNDED}_\mathbb{R} \in H\exists$ .

The proof of  $H\forall$ -hardness is based on the following lemma.

**Lemma 3.4.** *There exists a constant  $c > 0$  with the following property. To  $f \in \mathbb{R}[\varepsilon, X_1, \dots, X_n]$  of degree  $d$  and  $N = (nd)^{cn}$  we assign the semialgebraic set*

$$S := \left\{ (\varepsilon, x, y) \in (0, \infty) \times (-1, 1)^n \times \mathbb{R} \mid f(\varepsilon, x) = 0 \wedge y \prod_{k=1}^n (1 - x_k^2) = \varepsilon^N \right\}.$$

Then for all  $f$  we have

$$H\varepsilon \forall x \in (-1, 1)^n f(\varepsilon, x) \neq 0 \iff S \text{ is closed in } \mathbb{R}^{n+2}.$$

The proof of this lemma uses efficient quantifier elimination over  $\mathbb{R}$ , cf. [23, Part III], and the following auxiliary result, whose proof is based on the description of the half-branches of real algebraic curves by means of Puiseux series, cf. [2, §13].

**Lemma 3.5.** *Let  $T \subseteq (0, \infty) \times (0, \infty)$  be a semialgebraic set given by a Boolean combination of inequalities of polynomials of degree strictly less than  $d$  and let  $(0, 0) \in \bar{T}$ . Then there exists a sequence of points  $(t_\nu, \varepsilon_\nu)$  in  $T$  such that*

$$\lim_{\nu \rightarrow \infty} \frac{\varepsilon_\nu^d}{t_\nu} = 0.$$

**PROOF OF THEOREM 3.2.** It suffices to prove  $H\forall$ -hardness. Lemma 3.4 (plus the reduction in the proof of Proposition 3.1 to allow the variables  $x_i$  to vary in  $\mathbb{R}$ ) allows us to reduce  $\text{STANDARD}(H\forall)$  to  $\text{BASICCLOSED}_\mathbb{R}$ . Indeed, a description of the set  $S$  in its statement can be obtained in polynomial time from a description

of  $f$ . However, the exponent  $N$  is exponential in the size of  $f$ . In order to reduce the degree  $N$  we introduce the variables  $z_1, \dots, z_{\log N}$  (assuming  $N$  is a power of 2) and replace  $y \prod_{k=1}^n (1 - x_k^2) = \varepsilon^N$  by the equalities

$$z_1 = \varepsilon^2, \quad z_j = z_{j-1}^2 \quad (j = 2, \dots, \log N), \quad y \prod_{k=1}^n (1 - x_k^2) = z_{\log N}.$$

This defines a basic semialgebraic set  $S'$  homeomorphic to  $S$  whose size in dense encoding is polynomial in the size of  $f$ . This completes the proof for  $\text{BASICCLOSED}_{\mathbb{R}}$ . Hardness of  $\text{BASICCOMPACT}_{\mathbb{R}}$  follows as before by means of the stereographic projection.  $\square$

*Problem 3.6.* Can Theorem 3.2 be extended to arbitrary semialgebraic sets? We note that the three problems of deciding, for an arbitrary semialgebraic set  $S$ , whether  $S$  is compact, whether it is open, or whether it is closed are polynomial time equivalent.

Complexity results for problems involving functions instead of sets are also of interest. Consider the following problems:

- $\text{CONT}_{\mathbb{R}}$  (*Continuity*) Given a circuit  $\mathcal{C}$ , decide whether  $f_{\mathcal{C}}$  is total and continuous.
- $\text{CONT}_{\mathbb{R}}^{\text{DF}}$  (*Continuity for Division-Free Circuits*) Given a division-free circuit  $\mathcal{C}$ , decide whether  $f_{\mathcal{C}}$  is continuous.
- $\text{CONTPPOINT}_{\mathbb{R}}^{\text{DF}}$  (*Continuity at a Point for Division-Free Circuits*) Given a division-free circuit  $\mathcal{C}$  with  $n$  input gates and a point  $x \in \mathbb{R}^n$ , decide whether  $f_{\mathcal{C}}$  is continuous at  $x$ .

**Theorem 3.7.**  $\text{CONTPPOINT}_{\mathbb{R}}^{\text{DF}}$  is  $\text{H}\forall$ -complete. Moreover,  $\text{CONT}_{\mathbb{R}}^{\text{DF}} \in \text{H}^2\forall$  and  $\text{CONT}_{\mathbb{R}} \in \text{H}^3\forall$  and both problems are  $\forall$ -hard.

## 4 Quantifying Genericity

It is customary to express denseness in terms of adherence. For instance, a subset  $S \subseteq \mathbb{R}^n$  is Euclidean dense in  $\mathbb{R}^n$  iff  $\forall x \in \mathbb{R}^n (x, S) \in \text{EADH}_{\mathbb{R}}$ . We formally define  $\text{EDENSE}_{\mathbb{R}}$  as follows:

- $\text{EDENSE}_{\mathbb{R}}$  (*Euclidean Denseness*) Given a decision circuit  $\mathcal{C}$  with  $n$  input gates, decide whether  $\overline{S_{\mathcal{C}}} = \mathbb{R}^n$ .

Therefore, one would expect at least  $\text{NP}_{\mathbb{R}}$ -hardness (if not  $\Pi_{\mathbb{R}}^2$ -completeness) for  $\text{EDENSE}_{\mathbb{R}}$ . The situation is quite different, however. Let the problem  $\text{ZDENSE}_{\mathbb{R}}$  be the counterpart of  $\text{EDENSE}_{\mathbb{R}}$  for the Zariski topology.

**Proposition 4.1.**  $\text{EDENSE}_{\mathbb{R}}$  is  $\forall^*$ -complete and  $\text{ZDENSE}_{\mathbb{R}}$  is  $\exists^*$ -complete.

The following result locates  $\exists^*$  and  $\forall^*$  with respect to the previously studied complexity classes.

**Proposition 4.2.** *We have  $\exists^* \subseteq \exists \subseteq H^2\exists^*$  and  $\forall^* \subseteq \forall \subseteq H^2\forall^*$ .*

PROOF. The proof of the inclusion  $\exists^* \subseteq \exists$  relies on a technique by Koiran [22] developed for showing that  $\text{DIM}(d)$  is in  $\text{NP}_{\mathbb{R}}$ . Using this technique, one may in fact show the following general inclusion for any polynomial complexity class  $\mathcal{C}$

$$\exists^*\mathcal{C} \subseteq \exists\mathcal{C} \text{ and } \forall^*\mathcal{C} \subseteq \forall\mathcal{C}. \tag{6}$$

In order to show that  $\exists \subseteq H^2\exists^*$  note that for  $f \in \mathbb{R}[X_1, \dots, X_n]$  we have

$$\begin{aligned} \exists x f(x) = 0 &\iff H\delta \exists x (\|x\|^2 \leq \delta^{-1} \wedge f(x) = 0) \\ &\iff H\delta H\varepsilon \exists x (\|x\|^2 \leq \delta^{-1} \wedge f(x)^2 < \varepsilon) \\ &\iff H\delta H\varepsilon \exists^* x (\|x\|^2 < \delta^{-1} \wedge f(x)^2 < \varepsilon) \end{aligned}$$

the second equivalence by the compactness of closed balls. □

## 5 Discrete Setting

We discuss here the relationship between polynomial classes and classical complexity theory. Thus we restrict the input polynomials in the problems considered so far to polynomials with integer coefficients (represented in binary), or to constant-free circuits (i.e., circuits which use only 0 and 1 as values associated to their constant nodes). The resulting problems can be encoded in a finite alphabet and studied in the classical Turing setting. In general, if  $L$  denotes a problem defined over  $\mathbb{R}$  or  $\mathbb{C}$ , we denote its restriction to integer inputs by  $L^{\mathbb{Z}}$ . This way, the discrete problems  $\text{UNBOUNDED}_{\mathbb{R}}^{\mathbb{Z}}$ ,  $\text{EADH}_{\mathbb{R}}^{\mathbb{Z}}$ ,  $\text{BASICCLOSED}_{\mathbb{R}}^{\mathbb{Z}}$ , etc. are well defined.

Another natural restriction (considered e.g. in [13,20,21]), now for real machines, is the requirement that no constants other than 0 and 1 appear in the machine program. Complexity classes arising by considering such constant-free machines are indicated by a superscript 0 as in  $\text{P}_{\mathbb{R}}^0$ ,  $\text{NP}_{\mathbb{R}}^0$ , etc.

The simultaneous consideration of both these restrictions leads to the notion of constant-free Boolean part.

**Definition 5.1.** *Let  $\mathcal{C}$  be a complexity class over  $\mathbb{R}$ . The Boolean part of  $\mathcal{C}$  is the discrete complexity class*

$$\text{BP}(\mathcal{C}) = \{S \cap \{0, 1\}^{\infty} \mid S \in \mathcal{C}\}.$$

We denote by  $\mathcal{C}^0$  the subclass of  $\mathcal{C}$  obtained by requiring all the considered machines over  $\mathbb{R}$  to be constant-free. The constant-free Boolean part of  $\mathcal{C}$  is defined as  $\text{BP}^0(\mathcal{C}) := \text{BP}(\mathcal{C}^0)$ .



Some of the classes  $BP^0(\mathcal{C})$  do contain natural complete problems. This raises the issue of characterizing these classes in terms of already known discrete complexity classes. Unfortunately, there are not many real complexity classes  $\mathcal{C}$  for which  $BP^0(\mathcal{C})$  is completely characterized in such terms. The only such result we know is  $BP^0(\text{PAR}_{\mathbb{R}}) = \text{PSPACE}$ , proved in [12]. An obvious solution (which may be the only one) is to define new discrete complexity classes in terms of Boolean parts. In this way we define the classes  $\text{PR} := BP^0(\text{P}_{\mathbb{R}})$ ,  $\text{NPR} := BP^0(\text{NP}_{\mathbb{R}})$  and  $\text{coNPR} = \text{coBP}^0(\text{NP}_{\mathbb{R}}) = BP^0(\text{coNP}_{\mathbb{R}})$ .

While never explicitated as a complexity class, the computational resources behind PR have been around for quite a while. A constant-free machine over  $\mathbb{R}$  restricted to binary inputs is, in essence, a unit-cost Random Access Machine (RAM). Therefore, PR is the class of subsets of  $\{0, 1\}^*$  decidable by a RAM in polynomial time. In [1] it was shown that PR is contained in the counting hierarchy and some empirical evidence pointing towards  $\text{P} \neq \text{PR}$  was collected. We also note that the existential theory of the reals over the language  $\{\{0, 1\}, +, -, \times, \leq\}$  is an NPR-complete problem.

**Theorem 5.2.** *For any polynomial class  $\mathcal{C}$  we have  $BP^0(\text{HC}) = BP^0(\mathcal{C})$ .*

The proof is based on the old idea of simulating the infinitesimal  $\varepsilon$  by a doubly exponentially small number  $2^{2^{N^c}}$ , which can be computed by a straight-line program in time polynomial in  $N$  by repeated squaring. A second ingredient is the theorem on efficient quantifier elimination [23, Part III].

Combining Theorem 5.2 with Proposition 4.2 we obtain:

**Corollary 5.3.** *We have  $BP^0(\exists^*) = BP^0(\exists) = BP^0(\text{H}\exists) = \text{NPR}$  and  $BP^0(\forall^*) = BP^0(\forall) = BP^0(\text{H}\forall) = \text{coNPR}$ .*

All our completeness results induce completeness results in the classical setting.

**Corollary 5.4.** *(a) The discrete versions of  $\text{UNBOUNDED}_{\mathbb{R}}$ ,  $\text{EADH}_{\mathbb{R}}$ ,  $\text{LOC DIM}_{\mathbb{R}}$ , and  $\text{ZDENSE}_{\mathbb{R}}$  are NPR-complete.*

*(b) The discrete versions of the following problems are coNPR-complete:  $\text{BASICCLOSED}_{\mathbb{R}}$ ,  $\text{BASICCOMPACT}_{\mathbb{R}}$ ,  $\text{EDENSE}_{\mathbb{R}}$ ,  $\text{CONTPPOINT}_{\mathbb{R}}^{\text{DF}}$ ,  $\text{CONTR}_{\mathbb{R}}$ .*

**PROOF.** The claimed memberships follow from the definition of  $BP^0$ , Corollary 5.3, and a cursory look at the membership proofs for their real versions which show that the involved algorithms are constant-free.

For proving hardness we first note that  $\text{STANDARD}(\text{H}\exists)^{\mathbb{Z}}$  is hard for  $BP^0(\text{H}\exists)$  (and similarly for  $\text{STANDARD}(\exists^*)$ ). Indeed, when restricted to binary inputs, the reduction in the proof of Proposition 2.2 can be performed by a Turing machine in polynomial time. We next note that the reductions shown in this paper for all the problems above also can be performed by a Turing machine in polynomial time when restricted to binary inputs. □

Thus, based on Theorem 5.2, we obtain in Corollary 5.4 the completeness for the discrete problems  $\text{CONTPoint}_{\mathbb{R}}^{\text{DF}}$  and  $\text{CONTPoint}_{\mathbb{R}}^{\mathbb{Z}}$  even though we do not have completeness results for the corresponding real problems. This suggests that we are not far away from completeness.

## References

1. Allender, E., Bürgisser, P., Kjeldgaard-Pedersen, J., Bro-Miltersen, P.: On the complexity of numerical analysis. In: Proc. 21st Ann. IEEE Conference on Computational Complexity, pp. 331–339. IEEE Computer Society Press, Los Alamitos (2006)
2. Bliss, G.A.: Algebraic functions. Dover Publications, New York (1966)
3. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer, Heidelberg (1998)
4. Blum, L., Shub, M., Smale, S.: On a theory of computation and complexity over the real numbers. Bull. Amer. Math. Soc. 21, 1–46 (1989)
5. Bochnak, J., Coste, M., Roy, M.F.: Géométrie algébrique réelle, Folge. Ergebnisse der Mathematik und ihrer Grenzgebiete, 3, vol. 12. Springer, Heidelberg (1987)
6. Bournez, O., Cucker, F., de Naurois, P.J., Marion, J.-Y.: Implicit complexity over an arbitrary structure: sequential and parallel polynomial time. J. Logic Comput. 15(1), 41–58 (2005)
7. Bürgisser, P., Cucker, F.: Counting complexity classes for numeric computations I: Semilinear sets. SIAM J. Comp. 33, 227–260 (2003)
8. Bürgisser, P., Cucker, F.: Counting complexity classes for numeric computations II: Algebraic and semialgebraic sets. Journal of Complexity 22(2), 147–191 (2006)
9. Bürgisser, P., Cucker, F., de Naurois, P.J.: The complexity of semilinear sets in succinct representation. Comp. Compl. 15, 197–235 (2006)
10. Cucker, F.:  $\text{P}_{\mathbb{R}} \neq \text{NC}_{\mathbb{R}}$ . Journal of Complexity 8, 230–238 (1992)
11. Cucker, F.: On the Complexity of Quantifier Elimination: the Structural Approach. The Computer Journal 36, 399–408 (1993)
12. Cucker, F., Yu, D.: On the power of real Turing machines over binary inputs. SIAM J. Comp. 26, 243–254 (1997)
13. Cucker, F., Koiran, P.: Computing over the reals with addition and order: Higher complexity classes. Journal of Complexity 11, 358–376 (1995)
14. Cucker, F., Meer, K.: Logics which capture complexity classes over the reals. J. Symbolic Logic 64(1), 363–390 (1999)
15. Cucker, F., Rosselló, F.: On the complexity of some problems for the Blum, Shub & Smale model. In: Simon, I. (ed.) LATIN 1992. LNCS, vol. 583, pp. 117–129. Springer, Heidelberg (1992)
16. Cucker, F., Shub, M.: Generalized knapsack problems and fixed degree separations. Theoret. Comp. Sci. 161, 301–306 (1996)
17. Cucker, F., Torrecillas, A.: Two p-complete problems in the theory of the reals. Journal of Complexity 8, 454–466 (1992)
18. Grädel, E., Meer, K.: Descriptive complexity theory over the real numbers. In: Grädel, E., Meer, K. (eds.) The mathematics of numerical analysis, Park City, UT. Lectures in Appl. Math., vol. 32, pp. 381–403. Amer. Math. Soc, Providence, RI (1996)
19. Kleene, S.C.: Recursive predicates and quantifiers. Trans. Amer. Math. Soc. 53, 41–73 (1943)

20. Koiran, P.: Computing over the reals with addition and order. *Theoret. Comp. Sci.* 133, 35–47 (1994)
21. Koiran, P.: A weak version of the Blum, Shub & Smale model. *J. Comp. Syst. Sci.* 54, 177–189 (1997)
22. Koiran, P.: The real dimension problem is  $NP_R$ -complete. *Journal of Complexity* 15(2), 227–238 (1999)
23. Renegar, J.: On the computational complexity and geometry of the first-order theory of the reals. part I, II, III. *J. Symb. Comp.* 13(3), 255–352 (1992)
24. Larry, J.: The polynomial-time hierarchy. *Theoret. Comput. Sci.* 3(1), 1–22 (1976/1977)

# Online Conflict-Free Colorings for Hypergraphs<sup>\*</sup>

Amotz Bar-Noy<sup>1</sup>, Panagiotis Cheilaris<sup>1,2,3,\*\*</sup>, Svetlana Olonetsky<sup>4</sup>,  
and Shakhar Smorodinsky<sup>5,\*\*\*</sup>

<sup>1</sup> Brooklyn College

<sup>2</sup> The Graduate Center

City University of New York

<sup>3</sup> School of ECE

National Technical University of Athens

<sup>4</sup> Tel-Aviv University

<sup>5</sup> Courant Institute

New York University

**Abstract.** We provide a framework for online conflict-free coloring (CF-coloring) of any hypergraph. We use this framework to obtain an efficient randomized online algorithm for CF-coloring any  $k$ -degenerate hypergraph. Our algorithm uses  $O(k \log n)$  colors with high probability and this bound is asymptotically optimal for any constant  $k$ . Moreover, our algorithm uses  $O(k \log k \log n)$  random bits with high probability. As a corollary, we obtain asymptotically optimal randomized algorithms for online CF-coloring some hypergraphs that arise in geometry. Our algorithm uses exponentially fewer random bits compared to previous results.

We introduce deterministic online CF-coloring algorithms for points on the line with respect to intervals and for points on the plane with respect to halfplanes (or unit discs) that use  $\Theta(\log n)$  colors and recolor  $O(n)$  points in total.

## 1 Introduction

A *hypergraph* is a pair  $(V, \mathcal{E})$ , where  $V$  is a finite set and  $\mathcal{E} \subset 2^V$ . The set  $V$  is called the *ground set* or the *vertex set* and the elements of  $\mathcal{E}$  are called *hyperedges*. A *proper  $k$ -coloring* of a hypergraph  $H = (V, \mathcal{E})$ , for some positive integer  $k$ , is a function  $\chi : V \rightarrow \{1, 2, \dots, k\}$  such that no  $S \in \mathcal{E}$  with  $|S| \geq 2$  is monochromatic. Let  $\chi(H)$  denote the minimum integer  $k$  for which  $H$  has a  $k$ -coloring.  $\chi(H)$  is called the *chromatic number* of  $H$ . A *conflict-free coloring* (CF-coloring) of  $H$  is a coloring of  $V$  with the further restriction that for any hyperedge  $S \in \mathcal{E}$  there exists a vertex  $v \in S$  with a unique color (i.e., no other vertex of  $S$  has the same

---

<sup>\*</sup> The first two authors are partially supported by the CUNY Collaborative Incentive Research Grants Program Round 11 (2004–2006).

<sup>\*\*</sup> Supported by the European Social Fund (75%) and National Resources (25%) under the program EPEAEK II, ‘Heraclitus’.

<sup>\*\*\*</sup> Work on this paper was supported by the NSF Mathematical Sciences Postdoctoral Fellowship award 0402492.

color as  $v$ ). Both proper coloring and CF-coloring of hypergraphs are generalizations of vertex coloring of graphs (the definition coincides when the underlying hypergraph is a simple graph). Therefore the computational complexity of such colorings is at least as hard as for simple graphs.

The study of conflict-free colorings was originated in the work of Even et al. [6] and Smorodinsky [12] who were motivated by the problem of frequency assignment in cellular networks. Specifically, cellular networks are heterogeneous networks with two different types of nodes: *base stations* (that act as servers) and *clients*. Base stations are interconnected by an external fixed backbone network whereas clients are connected only to base stations. Connections between clients and base stations are implemented by radio links. Fixed frequencies are assigned to base stations to enable links to clients. Clients continuously scan frequencies in search of a base station with good reception. The fundamental problem of frequency assignment in such cellular networks is to assign frequencies to base stations so that every client, located within the receiving range of at least one station, can be served by some base station, in the sense that the client is located within the range of the station and no other station within its reception range has the same frequency (such a station would be in “conflict” with the given station due to mutual interference). The goal is to minimize the number of assigned frequencies (“colors”) since the frequency spectrum is limited and costly.

Let  $\mathcal{R}$  be a set of regions in the plane. For a point  $p \in \cup_{r \in \mathcal{R}} r$ , let  $r(p) = \{r \in \mathcal{R} \mid p \in r\}$ . Let  $H(\mathcal{R})$  denote the hypergraph  $(\mathcal{R}, \{r(p) \mid p \in \cup_{r \in \mathcal{R}} r\})$ . We say that  $H(\mathcal{R})$  is the hypergraph *induced* by  $\mathcal{R}$ . Even et al. [6] showed that any hypergraph induced by a family  $\mathcal{R}$  of  $n$  discs in the plane admits a CF-coloring with only  $O(\log n)$  colors and that this bound is tight in the worst case. In addition to the practical motivation, this new coloring model has drawn much attention of researchers through its own theoretical interest and such colorings have been the focus of several recent papers (see, e.g., [1,5,6,7,8,10,11,12,13]). To capture a dynamic scenario where antennas can be added to the network, Chen et al. [7] initiated the study of *online* CF-coloring of hypergraphs. They considered a very simple hypergraph  $H$  which has its vertex set represented as a set  $P$  of  $n$  points on the line and its hyperedge set consists of all intersections of the points with some interval. The set  $P \subset \mathbb{R}$  is revealed by an adversary online: Initially,  $P$  is empty, and the adversary inserts points into  $P$ , one point at a time. Let  $P(t)$  denote the set  $P$  after the  $t$ -th point has been inserted. Each time a point is inserted, the algorithm needs to assign a color  $c(p)$  to it, which is a positive integer. Once the color has been assigned to  $p$ , it cannot be changed in the future. The coloring should remain conflict-free at all times. That is, for any interval  $I$  that contains points of  $P(t)$ , there is a color that appears exactly once in  $I$ . Among other results, [7] provided a deterministic algorithm for online CF-coloring  $n$  points on the line with  $\Theta(\log^2 n)$  colors in the worst case.

*An online CF-coloring framework:* In this paper, we investigate the most general form of online CF-coloring applied to arbitrary hypergraphs. Suppose the vertices of an underlying hypergraph  $H = (V, \mathcal{E})$  are given online by an adversary. Namely, at every given time step  $t$ , a new vertex  $v_t \in V$  is given and

the algorithm must assign  $v_t$  a color such that the coloring is a valid conflict-free coloring of the hypergraph that is induced by the vertices  $V_t = \{v_1, \dots, v_t\}$  (see the exact definition in section 2). Once  $v_t$  is assigned a color, that color cannot be changed in the future. The goal is to find an algorithm that minimizes the maximum total number of colors used (where the maximum is taken over all permutations of the set  $V$ ).

We present a general framework for online CF-coloring any hypergraph. Interestingly, this framework is a generalization of some known coloring algorithms. For example the Unique-Max Algorithm of [7] can be described as a special case of our framework. Also, when the underlying hypergraph is a simple graph then the First-Fit online algorithm is another special case of our framework. Based on this framework, we introduce a *randomized algorithm* and show that the maximum number of colors used is a function of the ‘degeneracy’ of the hypergraph. We define the notion of a  $k$ -degenerate hypergraph as a generalization of the same notion for simple graphs. Specifically we show that if the hypergraph is  $k$ -degenerate, then our algorithm uses  $O(k \log n)$  colors with high probability. This is asymptotically tight for any constant  $k$ .

As demonstrated in [7], the problem of online CF-coloring the very special hypergraph induced by points on the real line with respect to intervals is highly non-trivial. Chen, Kaplan and Sharir [10] studied the special hypergraph induced by points in the plane with respect to halfplanes and unit discs and obtained a randomized online CF-coloring with  $O(\log^3 n)$  colors with high probability. Recently, the (randomized) bound  $\Theta(\log n)$  just for these two special cases was obtained independently by Chen [4]. Our algorithm is more general and uses only  $\Theta(\log n)$  colors; an interesting evidence to our algorithm being fundamentally different from the ones in [4,7,10], when used for the special case of hypergraphs that arise in geometry, is that it uses exponentially fewer random bits. The algorithms of [4,7,10] use  $\Theta(n \log n)$  random bits, whereas our algorithm uses  $O(\log n)$  random bits.

Another interesting corollary of our result is that we obtain a randomized online coloring for  $k$ -inductive graphs with  $O(k \log n)$  colors with high probability. This case was studied by Irani [9] who showed that a greedy First-Fit algorithm achieves the same bound deterministically.

*Deterministic online CF-coloring with recoloring:* We initiate the study of online CF-coloring where at each step, in addition to the assignment of a color to the newly inserted point, we allow some recoloring of other points. The bi-criteria goal is to minimize the total number of recolorings done by the algorithm and the total number of colors used by the algorithm. We introduce an online algorithm for CF-coloring points on the line with respect to intervals, where we recolor at most one already assigned point at each step. Our algorithm uses  $\Theta(\log n)$  colors. This is in contrast with the  $O(\log^2 n)$  colors used by the best known deterministic algorithm by Chen et al. [7] that does not recolor points. We also show online algorithm for CF-coloring points on the plane with respect to halfplanes that uses  $\Theta(\log n)$  colors and the total number of recolorings is  $O(n)$ . For this problem no deterministic algorithm was known before.

*Paper organization:* Section 2 defines the notion of a  $k$ -degenerate hypergraph. Section 3 presents the general framework for online CF-coloring of hypergraphs. Section 4 introduces the randomized algorithm derived from the framework. Section 5 shows deterministic online algorithm for intervals and halfplanes with recoloring. Section 6 describes the results for the hypergraphs that arise from geometry. Finally, Section 7 concludes with a discussion and some open problems.

## 2 Preliminaries

We start with some basic definitions:

**Definition 1.** Let  $H = (V, \mathcal{E})$  be a hypergraph. For a subset  $V' \subset V$ , let  $H(V')$  be the hypergraph  $(V', \mathcal{E}')$  where  $\mathcal{E}' = \{e \cap V' \mid e \in \mathcal{E} \text{ and } e \cap V' \neq \emptyset\}$ .  $H(V')$  is called the induced hypergraph on  $V'$ .

**Definition 2.** For a hypergraph  $H = (V, \mathcal{E})$ , the Delaunay graph  $G(H)$  is the simple graph  $G = (V, E)$  where the edge set  $E$  is defined as  $E = \{(x, y) \mid \{x, y\} \in \mathcal{E}\}$  (i.e.,  $G$  is the graph on the vertex set  $V$  whose edges consist of all hyperedges in  $H$  of cardinality two).

**Definition 3.** A simple graph  $G = (V, E)$  is called  $k$ -degenerate (or  $k$ -inductive) for some positive integer  $k$ , if every (vertex-induced) subgraph of  $G$  has a vertex of degree at most  $k$ .

**Definition 4.** Let  $k > 0$  be a fixed integer and let  $H = (V, \mathcal{E})$  be a hypergraph on  $n$  vertices. Fix a subset  $V' \subset V$ . For a permutation  $\pi$  of  $V'$  such that  $V' = \{v_1, \dots, v_i\}$  (where  $i = |V'|$ ) let  $C_\pi(V') = \sum_{j=1}^i d(v_j)$ , where  $d(v_j) = |\{l < j \mid (v_j, v_l) \in G(H(\{v_1, \dots, v_j\}))\}|$ , that is,  $d(v_j)$  is the number of neighbors of  $v_j$  in the Delaunay graph of the hypergraph induced by  $\{v_1, \dots, v_j\}$ . Assume that  $\forall V' \subset V$  and for all permutations  $\pi \in S_{|V'|}$  we have  $C_\pi(V') \leq k|V'|$ . Then we say that  $H$  is  $k$ -degenerate.

It is not difficult to see that our definition of a  $k$ -degenerate hypergraph is a generalization of that of a  $k$ -degenerate graph.

## 3 An Online CF-Coloring Framework

Let  $H = (V, \mathcal{E})$  be any hypergraph. Our goal is to define a framework that colors the vertices  $V$  in an online fashion. That is, the vertices of  $V$  are revealed by an adversary one at a time. At each time step  $t$ , the algorithm must assign a color to the newly revealed vertex  $v_t$ . This color cannot be changed in the future. The coloring has to be conflict-free for all the induced hypergraphs  $H(V_t)$   $t = 1, \dots, n$ , where  $V_t \subset V$  is the set of vertices revealed by time  $t$ .

For a fixed positive integer  $h$ , let  $A = \{a_1, \dots, a_h\}$  be a set of  $h$  auxiliary colors (not to be confused with the set of ‘real’ colors used for the CF-coloring:  $\{1, 2, \dots\}$ ). Let  $f : \mathbb{N} \rightarrow A$  be some fixed function. We now define the framework that depends on the choice of the function  $f$  and the parameter  $h$ .

A table (to be updated online) is maintained where each entry  $i$  at time  $t$  is associated with a subset  $V_t^i \subset V_t$  in addition to an auxiliary proper coloring of  $H(V_t^i)$  with at most  $h$  colors. We say that  $f(i)$  is the color that represents entry  $i$  in the table. At the beginning all entries of the table are empty. Suppose all entries of the table are updated until time  $t - 1$  and let  $v_t$  be the vertex revealed by the adversary at time  $t$ . The framework first checks if an auxiliary color can be assigned to  $v_t$  such that the auxiliary coloring of  $V_{t-1}^1$  together with the color of  $v_t$  is a proper coloring of  $H(V_{t-1}^1 \cup \{v_t\})$ . Any (proper) coloring procedure can be used by the framework. For example a first-fit greedy in which all colors in the order  $a_1, \dots, a_h$  are checked until one is found. If such a color cannot be found for  $v_t$ , then entry 1 is left with no changes and the process continues to the next entry. If however, such a color can be assigned, then  $v_t$  is added to the set  $V_{t-1}^1$ . Let  $c$  denote such an auxiliary color assigned to  $v_t$ . If this color is the same as  $f(1)$  (the auxiliary color that is associated with entry 1), then the final color in the online CF-coloring of  $v_t$  is 1 and the updating process for the  $t$ -th vertex stops. Otherwise, if an auxiliary color cannot be found or if the assigned auxiliary color is not the same as the color associated with this entry, the updating process continues to the next entry. The updating process stops at the first entry  $i$  for which  $v_t$  is both added to  $V_t^i$  and the auxiliary color assigned to  $v_t$  is the same as  $f(i)$ . The color of  $v_t$  in the final conflict-free coloring is then set to  $i$ .

It is possible that  $v_t$  never gets a final color. In this case we say that the framework does not halt. However, termination can be guaranteed by imposing some restrictions on the auxiliary coloring method and the choice of the function  $f$ . For example, if first-fit is used for the auxiliary colorings at any entry and if  $f$  is the constant function  $f(i) = a_1$ , for all  $i$ , then the framework is guaranteed to halt for any time  $t$ . In section 4 we derive a randomized online algorithm based on this framework. This algorithm halts with probability 1 and moreover it halts after a “small” number of entries with high probability. The above framework produces a valid CF-coloring in case it halts (proof omitted):

**Lemma 1.** *If the above framework halts for any vertex  $v_t$  then it produces a valid online CF-coloring of  $H$ .*

The above algorithmic framework can also describe some well-known deterministic algorithms. For example, if first-fit is used for auxiliary colorings and  $f$  is the constant function,  $f(i) = a_1$ , for all  $i$ , then: (a) If the input hypergraph is induced by points on a line with respect to intervals then the algorithm derived from the framework becomes identical to the Unique Maximum Greedy algorithm described and analyzed in [7]. (b) If the input is a  $k$ -degenerate graph (also called  $k$ -inductive graph), the derived algorithm is identical to the First-Fit greedy algorithm for coloring graphs online.



## 4 An Online Randomized CF-Coloring Algorithm

There is a randomized online CF-coloring in the oblivious adversary model that always produces a valid coloring and the number of colors used is related to the degeneracy of the underlying hypergraph in a manner described in theorem [1](#).

**Theorem 1.** *Let  $H = (V, \mathcal{E})$  be a  $k$ -degenerate hypergraph on  $n$  vertices. Then there exists a randomized online CF-coloring algorithm for  $H$  which uses at most  $O(\log_{1+\frac{1}{4k+1}} n) = O(k \log n)$  colors with high probability.*

The algorithm is based on the framework of section [3](#). In order to define the algorithm, we need to state what is the function  $f$ , the set of auxiliary colors of each entry and the algorithm we use for the auxiliary coloring at each entry. We use the set  $A = \{a_1, \dots, a_{2k+1}\}$ . For each entry  $i$ , the representing color  $f(i)$  is chosen uniformly at random from  $A$ . We use a first-fit algorithm for the auxiliary coloring.

Our assumption on the hypergraph  $H$  (being  $k$ -degenerate) implies that at least half of the vertices up to time  $t$  that ‘reached’ entry  $i$  (but not necessarily added to entry  $i$ ), denoted by  $X_{\geq i}^t$ , have been actually given some auxiliary color at entry  $i$  (that is,  $|V_t^i| \geq \frac{1}{2} |X_{\geq i}^t|$ ). This is easily implied by the fact that at least half of those vertices  $v_t$  have at most  $2k$  neighbors in the Delaunay graph of the hypergraph induced by  $X_{\geq i}^{t-1}$  (since the sum of these quantities is at most  $k |X_{\geq i}^t|$  and since  $V_t^i \subset X_{\geq i}^t$ ). Therefore since we have  $2k + 1$  colors available, there is always a free color to assign to such a vertex. The following lemma shows that if we use one of these ‘free’ colors then the updated coloring is indeed a proper coloring of the corresponding induced hypergraph as well (proof omitted).

**Lemma 2.** *Let  $H = (V, \mathcal{E})$  be a  $k$ -degenerate hypergraph and let  $V_t^j$  be the subset of  $V$  at time  $t$  and at level  $j$  as produced by the above algorithm. Then for any  $j$  and  $t$  if  $v_t$  is assigned a color distinct from all its neighbors in the Delaunay graph  $G(H(V_t^j))$  then this color together with the colors assigned to the vertices  $V_{t-1}^j$  is also a proper coloring of the hypergraph  $H(V_t^j)$ .*

We proceed to the analysis of the performance of the algorithm.

**Lemma 3.** *Let  $H = (V, \mathcal{E})$  be a hypergraph and let  $\chi$  be a coloring produced by the above algorithm on an online input  $V = \{v_t\}$  for  $t = 1, \dots, n$ . Let  $X_i$  (respectively  $X_{\geq i}$ ) denote the random variable counting the number of points of  $V$  that were assigned a final color at entry  $i$  (respectively a final color at some entry  $\geq i$ ). Let  $\mathbf{E}_i = \mathbf{E}[X_i]$  and  $\mathbf{E}_{\geq i} = \mathbf{E}[X_{\geq i}]$  (note that  $X_{\geq i+1} = X_{\geq i} - X_i$ ). Then:*

$$\mathbf{E}_{\geq i} \leq \left( \frac{4k+1}{4k+2} \right)^{i-1} n$$

*Proof.* By induction on  $i$ . The case  $i = 1$  is trivial. Assume that the statements hold for  $i$ . To complete the induction step, we need to prove that  $\mathbf{E}_{\geq i+1} \leq \left( \frac{4k+1}{4k+2} \right)^i n$ .

By the conditional expectation formula, we have for any two random variables  $X, Y$  that  $\mathbf{E}[X] = \mathbf{E}[\mathbf{E}[X \mid Y]]$ . Thus

$$\mathbf{E}_{\geq i+1} = \mathbf{E}[\mathbf{E}[X_{\geq i+1} \mid X_{\geq i}]] = \mathbf{E}[\mathbf{E}[X_{\geq i} - X_i \mid X_{\geq i}]] = \mathbf{E}[X_{\geq i} - \mathbf{E}[X_i \mid X_{\geq i}]].$$

It is easily seen that  $\mathbf{E}[X_i \mid X_{\geq i}] \geq \frac{1}{2} \frac{X_{\geq i}}{2k+1}$  since at least half of the vertices of  $X_{\geq i}$  got an auxiliary color by the above algorithm. Moreover each of those elements that got an auxiliary color had probability  $\frac{1}{2k+1}$  (This is the only place where we need to assume that the adversary is oblivious and does not have access to the random bits) to get the final color  $i$ . Thus

$$\mathbf{E}[X_{\geq i} - \mathbf{E}[X_i \mid X_{\geq i}]] \leq \mathbf{E}[X_{\geq i} - \frac{1}{2(2k+1)} X_{\geq i}] = \frac{4k+1}{4k+2} \mathbf{E}[X_{\geq i}] \leq \left(\frac{4k+1}{4k+2}\right)^i n$$

by linearity of expectation and by the induction hypotheses. □

**Lemma 4.** *The expected number of colors used by the above algorithm is at most  $\log_{\frac{4k+2}{4k+1}} n + 1$ .*

*Proof.* Let  $I_i$  be the indicator random variable for the following event: some points are colored with a real color in entry  $i$ . We are interested in the number of colors used, that is  $Y := \sum_{i=1}^{\infty} I_i$ . Let  $b(k, n) = \log_{(4k+2)/(4k+1)} n$ . Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{1 \leq i} I_i\right] = \mathbf{E}\left[\sum_{1 \leq i \leq b(k, n)+1} I_i\right] + \mathbf{E}[X_{\geq b(k, n)+1}] \leq b(k, n) + 1$$

and we also have the following concentration result:

$$\Pr[\text{more than } c \cdot b(k, n) \text{ colors are used}] \leq \mathbf{E}_{\geq c \cdot b(k, n)+1} \leq \frac{1}{n^{c-1}}$$

both by Markov’s inequality and lemma 3. □

*Remark:* In the above description of the algorithm, all the random bits are chosen in advance (by deciding the values of the function  $f$  in advance). However, one can be more efficient and calculate the entry  $f(i)$  only at the first time we need to update entry  $i$ , for any  $i$ . Since at each entry we need to use  $O(\log k)$  random bits and we showed that the number of entries used is  $O(k \log n)$  w.h.p then the total number of random bits used by our algorithm is  $O(k \log k \log n)$  w.h.p.

## 5 Deterministic Online CF-Coloring with Recoloring

### 5.1 An $O(\log n)$ Algorithm with Recoloring for Intervals

We describe a deterministic online CF-coloring algorithm for intervals that is only allowed to recolor a single “old” point during each insertion of a new point. The algorithm is based on the framework developed in Section 3 where we use 3 auxiliary colors  $\{a, b, c\}$  and  $f$  is the constant function  $f(l) = a \ \forall l$ . We refer to

points colored with  $b$  or  $c$  as  $d$ -points. In order to have only logarithmic number of entries, we slightly modify the framework (using a recoloring procedure) such that the number of points colored with  $a$  in each entry of the table is at least one third of the total points that reach that entry. To achieve this goal, our algorithm maintains the following invariant in every level: There are at most two  $d$ -points between every pair of points colored with  $a$  (i.e., between every pair that are consecutively colored  $a$  among the  $a$ -points). Therefore, at least a third of the points at each entry get color  $a$ , and two thirds are deferred for coloring in a higher entry. The total number of colors is at most  $\log_{3/2} n + 1$ . When a new point  $p$  arrives, it is colored according to the following algorithm, starting from entry 1:

- If  $p$  is not adjacent to a point colored with an auxiliary color  $a$  then  $p$  is assigned auxiliary color  $a$  and gets its final color in that entry.
- We color point  $p$  with  $b$  or  $c$  greedily as long as it does not break the invariant that between any two consecutive  $a$ 's we have at most two  $d$ -points.
- It remains to handle the case where the new point  $p$  has a point colored with  $a$  on one side and a point, say  $q$ , colored with  $d$  on the other side, such that  $q$  has no adjacent point colored with  $a$ . We assign to  $p$  the auxiliary color of  $q$  (thus it is a  $d$ -point) in the current entry and in all higher entries for which  $q$  got an auxiliary color and assign to it the real color of  $q$ , and we recolor  $q$  with the auxiliary color  $a$  (and delete the corresponding appearance of it in all higher entries of the table), and thus we recolor  $q$  with the real color of the current entry. At this point all points have an assignment of real colors. It is not hard to check that when we recolor a point then we do not violate the invariants at any entry: Let  $\ell$  be the entry that caused recoloring, all entries before it remain the same, the change in the entry  $\ell$  does not break invariants, all other entries remain the same except that point  $p$  appears there instead of point  $q$  that was there before and there are no points between  $p$  and  $q$  that appear in an entry higher than  $\ell$ .

It can be easily checked that this algorithm produces a valid CF-coloring. Also, it can be proved that the number of recolorings is at most  $n - (\lceil \lg n \rceil + 1)$ , and this is tight.

## 5.2 An $O(\log n)$ Recoloring Algorithm for Circular Arcs

We define a hypergraph  $H$  closely related to the one induced by intervals: The vertex set of  $H$  is represented as a set  $P$  of  $n$  distinct points on a *circle*  $C$  and its hyperedge set consists of all intersections of the points with some *circular arc* of  $C$ . In the static case, it can be shown that  $n$  points can be optimally conflict-free colored with respect to circular arcs with  $\lfloor \lg(n-1) \rfloor + 2$  colors. Here, we are interested in an online setting, where the set  $P \subset C$  is revealed incrementally by an adversary, and, as usual, the algorithm has to commit on a color for each point without knowing how future points will be requested. Algorithms for intervals can be used almost verbatim for circular arcs. In fact,

the recoloring algorithm for intervals, given in section 5.1, can be used verbatim, if the notion of adjacency of points is adapted to the closed curve setting (for  $n \geq 3$ , each point has exactly 2 immediate neighboring points, whereas in the intervals case, the two extreme points have only one neighbor). Again, in each entry  $\ell$ , at least a third of the points is assigned auxiliary color  $a$ , and thus at most  $\log_{3/2} n + 1$  colors are used.

### 5.3 An $O(\log n)$ Algorithm for Circular Arcs with Substitution

We consider a variation on the problem of online conflict-free coloring with respect to circular arcs that was given in section 5.2. In this new variation, the adversary has, in addition to the ‘insertion move’ of a new point, a ‘substitution move’:

The adversary can substitute a set  $Q$  of already requested *consecutive* points with a single new point  $p$ , and the algorithm has to color  $p$ , such that the whole set of points is conflict-free colored with respect to circular arcs (in that set,  $p$  is included, but all points in  $Q$  are removed).

Our algorithm for this variation of the problem relies on the one given in section 5.2. For an insertion move of the adversary, it colors the new point like in section 5.2. For a substitution move of the adversary, it colors the new point  $p$ , with the *highest* color occurring in the points of  $Q$ . Point  $p$  also gets the entries of the unique point  $q \in Q$  with the highest color. It is not difficult to see that the coloring remains conflict-free after each move. We remark that a recoloring can happen only in an insertion move and that substitution moves do not make the algorithm introduce new colors. The following is true for every  $t$  (proof omitted):

**Lemma 5.** *After  $t$  moves, the above coloring algorithm uses at most  $\log_{3/2} t + 1$  colors.*

### 5.4 An $O(\log n)$ Algorithm with Recoloring for Halfplanes

In this section we describe a deterministic algorithm for online CF-coloring points with respect to halfplanes that uses  $O(\log n)$  colors and recolors  $O(n)$  points. Thus it can also be modified for CF-coloring points in the plane with respect to unit discs as remarked in Section 6. At every time instance  $t$ , the algorithm maintains the following invariant ( $V_t$  is the set of points that have appeared): All points (strictly) inside the convex hull of  $V_t$  are colored with the same special color, say ‘ $\star$ ’. The set of points on the convex hull of  $V_t$ , denoted by  $\text{CH}(V_t)$ , are colored with another set of colors, such that every set of consecutive points on the convex hull has a point with a unique color. If one considers the points of  $\text{CH}(V_t)$  in the circular order they appear on the convex hull, it is enough to CF-color them (with that circular order) with respect to circular arcs. The number of colors used in  $\text{CH}(V_t)$  must be logarithmic on  $t$ . It is not difficult to see that every subset of points of  $V_t$  induced by a halfplane contains a set of consecutive points on the convex hull of  $V_t$ , and thus the whole coloring is conflict-free.

We describe how the algorithm maintains the above invariant. A new point  $v_{t+1}$  that appears at time  $t+1$  is colored as follows: If it is inside the convex hull of  $V_t$ , then it gets color ‘ $\star$ ’. Otherwise, the new point  $v_{t+1}$  will be on  $\text{CH}(V_{t+1})$ , in which case we essentially use the algorithm of section 5.3 to color it. We have two cases, which correspond to a substitution and an insertion move, respectively:

- It might be the case that  $v_{t+1}$  forces some points (say they comprise set  $Q$ ) that were in  $\text{CH}(V_t)$  to belong to the interior of  $\text{CH}(V_{t+1})$ , so in order to maintain the invariant, all points in  $Q$  are recolored to ‘ $\star$ ’, and  $v_{t+1}$  is colored with the maximum color occurring in  $Q$  (this is like a substitution move of section 5.3).
- If, on the other hand, no points of  $\text{CH}(V_t)$  are forced into the convex hull, then point  $v_{t+1} \in \text{CH}(V_{t+1})$  is colored like in an insertion move of section 5.3 with the algorithm for circular arcs. In that last case, in order to maintain logarithmic number of colors on  $t$ , one recoloring of a point in  $\text{CH}(V_{t+1})$  might be needed.

The total number of recolorings is guaranteed to be  $O(n)$ , because for every insertion, at most one recoloring happens on the new convex hull, and every point colored with ‘ $\star$ ’ remains with that color.

## 6 Application to Geometry

Our randomized algorithm has applications to conflict-free colorings of certain geometric hypergraphs studied in [4,7,10]. We obtain the same asymptotic result as in [4] but with better constant of proportionalities and much fewer random bits. For example, if the hypergraph  $H$  is induced by intervals, it can be proved (with an analysis similar to the one given in section 4) that for any order of insertion of  $n$  points, when the auxiliary color for each entry is chosen uniformly at random from  $\{a, b, c\}$ , the expected number of colors used is bounded by  $\log_{\frac{3}{2}} n + 1$ . It is interesting that the best known upper bound for dynamically coloring  $n$  points deterministically, when the whole insertion order is known in advance, is also  $\log_{\frac{3}{2}} n + 1$  (see [3] for further details). In our algorithm the expected number of colors is bounded by  $1 + \log_{3/2} n \approx 1.71 \log_2 n$ , whereas in [4] by  $1 + \log_{8/7} n \approx 5.19 \log_2 n$ , three times our bound. When  $H$  is the hypergraph obtained by points in the plane intersected by halfplanes or unit disks, we obtain online randomized algorithms that use  $O(\log n)$  colors w.h.p. We summarize it as follows:

**Lemma 6.** *Let  $V$  be a finite set of  $n$  points in the plane and let  $\mathcal{E}$  be all subsets of  $V$  that can be obtained by intersecting  $V$  with a halfplane. Then the hypergraph  $H = (V, \mathcal{E})$  is 4-degenerate.*

*Proof.* The proof uses a few geometric lemmata. Details are omitted. □

**Corollary 1.** *Let  $H$  be the hypergraph as in lemma 6. Then the expected number of colors used by our randomized online CF-coloring applied to  $H$  is at most*

$\log_{\frac{18}{17}} n + 1$ . Also the actual number of colors used is  $O(\log_{\frac{18}{17}} n)$  with high probability. The number of random bits is  $O(\log n)$  with high probability

*Proof.* The proof follows immediately from lemmata [6, 4] and theorem [1]. □

**Corollary 2.** *Let  $V$  be a finite set of  $n$  points in the plane and let  $\mathcal{E}$  be all subsets of  $V$  that can be obtained by intersecting  $V$  with a unit disc. Then there exists a randomized online algorithm for CF-coloring  $H$  which uses  $O(\log n)$  colors and  $O(\log n)$  random bits with high probability.*

*Proof (outline).* Chen, Kaplan and Sharir [10] observed that by an appropriate partitioning of the plane one can modify any online algorithm for CF-coloring points with respect to halfplanes to a CF-coloring of points with respect to congruent discs. Using the same technique as developed in [10] and Corollary [1] we obtain the desired result. □

## 7 Discussion and Open Problems

We presented a framework for online CF-coloring any hypergraph. This framework coincides with some known algorithms in the literature when restricted to some special underlying hypergraphs. We derived a randomized online algorithm for CF-coloring any hypergraph (in the oblivious adversary model) and showed that the performance of our algorithm depends on a parameter which we refer to as the degeneracy of the hypergraph which is a generalization of the known notion of degeneracy in graphs (i.e., when the hypergraph is a simple graph then this term is the same as the classical definition of degeneracy of a graph; see Definition [3]). Specifically, if the hypergraph is  $k$ -degenerate then our algorithm uses  $O(k \log n)$  colors w.h.p, which is asymptotically optimal for any constant  $k$ , and  $O(k \log k \log n)$  random bits. This is the first efficient online CF-coloring for general hypergraphs and subsumes all the previous randomized algorithmic results of [4,7,10]. It also substantially improves the efficiency with respect to the amount of randomness used in the special cases studied in [4,7,10].

It would be interesting to find an efficient online deterministic algorithm for conflict-free coloring  $k$ -degenerate hypergraphs. Even for the very special case of a hypergraph induced by points and intervals where the number of neighbors of the Delaunay graph of every induced hypergraph is at most two), the best known deterministic online CF-coloring algorithm from [7] uses  $\Theta(\log^2 n)$  colors. We hope that our technique together with a possible clever de-randomization technique can shed light on this problem.

As mentioned already, the framework of section [3] can describe some known algorithms such as the Unique Max Greedy of [7] for online CF-coloring points on a line with respect to intervals. No sharp asymptotic bounds are known for the performance of Unique Max Greedy. The best known upper bound is linear, whereas the best known lower bound is  $\Omega(\sqrt{n})$ . We believe that this new approach could help analyze the performance of Unique Max Greedy.

In Section [5] we initiate the study of online CF-coloring with recoloring: We provide a deterministic online CF-coloring for points on the real line with respect

to intervals and show that our algorithm uses  $\Theta(\log n)$  colors and at most one recoloring per insertion. This is in contrast with the best known deterministic online CF-coloring for this case that uses  $\Theta(\log^2 n)$  colors in the worst case and does not recolor any other point ([7]). We also present deterministic online algorithms for CF-coloring points with respect to circular arcs and halfplanes (and unit discs) that use  $O(\log n)$  colors and  $O(n)$  recolorings in the worst case. It would be interesting to obtain an online CF-coloring algorithm with  $O(\log n)$  and  $O(1)$  recolorings per insertion for the case of halfplanes.

*Acknowledgements.* We would like to thank Amos Fiat, Haim Kaplan, János Pach, David Peleg, and Micha Sharir for helpful discussions concerning the problems studied in this paper.

## References

1. Ajwani, D., Elbassioni, K., Govindarajan, S., Ray, S.: Conflict-free coloring for rectangle ranges using  $O(n^{0.382+\epsilon})$  colors. In: 19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2007) (to appear)
2. Alon, N., Smorodinsky, S.: Conflict-free colorings of shallow discs. In: Proc. 22nd Annual ACM Symposium on Computational Geometry (SoCG 2006), pp. 41–43. ACM Press, New York (2006)
3. Bar-Noy, A., Cheilaris, P., Smorodinsky, S.: Conflict-free coloring for intervals: from offline to online. In: Proc. 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2006), pp. 128–137. ACM Press, New York (2006)
4. Chen, K.: On how to play a coloring game against color-blind adversaries. In: Proc. 22nd Annual ACM Symposium on Computational Geometry (SoCG 2006), pp. 44–51. ACM Press, New York (2006)
5. Elbassioni, K., Mustafa, N.: Conflict-Free Colorings of Rectangles Ranges. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 254–263. Springer, Heidelberg (2006)
6. Even, G., Lotker, Z., Ron, D., Smorodinsky, S.: Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM Journal on Computing* 33(1), 94–136 (2003) Also in Proceedings of the 43th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002)
7. Chen, K., Fiat, A., Kaplan, H., Levy, M., Matoušek, J., Mossel, E., Pach, J., Sharir, M., Smorodinsky, S., Wagner, U., Welzl, E.: Online conflict-free coloring for intervals. *SIAM Journal on Computing* 36, 1342–1359 (2006)
8. Har-Peled, S., Smorodinsky, S.: On conflict-free coloring of points and simple regions in the plane. *Discrete and Computational Geometry* 34, 47–70 (2005)
9. Irani, S.: Coloring inductive graphs on-line. *Algorithmica* 11(1), 53–72 (1994)
10. Chen, K., Kaplan, H., Sharir, M.: Online CF coloring for halfplanes, congruent disks, and axis-parallel rectangles. Manuscript (2005)
11. Pach, J., Tóth, G.: Conflict free colorings. In: *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*, Springer, Heidelberg (2003)
12. Smorodinsky, S.: *Combinatorial Problems in Computational Geometry*. Ph.D Dissertation, School of Computer Science, Tel-Aviv University (2003)
13. Smorodinsky, S.: On the chromatic number of some geometric hypergraphs. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006), pp. 316–323. ACM Press, New York (2006)

# Distributed Computing with Advice: Information Sensitivity of Graph Coloring

Pierre Fraigniaud<sup>1</sup>, Cyril Gavoille<sup>2</sup>, David Ilcinkas<sup>3,\*</sup>, and Andrzej Pelc<sup>3,\*\*</sup>

<sup>1</sup> CNRS and University Paris 7

<sup>2</sup> LaBRI, Université Bordeaux 1

<sup>3</sup> Département d'informatique, Université du Québec en Outaouais

**Abstract.** We study the problem of the amount of information (advice) about a graph that must be given to its nodes in order to achieve fast distributed computations. The required size of the advice enables to measure the information sensitivity of a network problem. A problem is information sensitive if little advice is enough to solve the problem rapidly (i.e., much faster than in the absence of any advice), whereas it is information insensitive if it requires giving a lot of information to the nodes in order to ensure fast computation of the solution. In this paper, we study the information sensitivity of distributed graph coloring.

## 1 Introduction

This work is a part of a recent project aiming at studying the quantitative impact of *knowledge* on the *efficiency* when computing with distributed entities (nodes of a distributed system, mobile users in ad hoc networks, etc.). Indeed, as observed by Linial [16], "within the various computational models for parallel computers, the limitations that follow from the local nature of the computation are specific to the distributed context". Two frameworks have been considered for analyzing the limitations incurring because of the local nature of the distributed computation. One aims at identifying which tasks can or cannot be computed locally, i.e., when every node can acquire knowledge only about the nodes that are at constant distance from it. Surprisingly, non trivial tasks can be achieved locally [20]. This is for instance the case of weak-coloring, a basis for a solution to some resource allocation problems. However, many important problems in distributed computing do not have a local solution [14]. This is the case of computing an approximate minimum vertex cover or an approximate minimum dominating set.

The other framework that has been considered is distributed computing *with advice*. In this framework, the computing entities can be given information about

---

\* A part of this work was done during the stay of this author at the Research Chair in Distributed Computing of the Université du Québec en Outaouais, as a postdoctoral fellow.

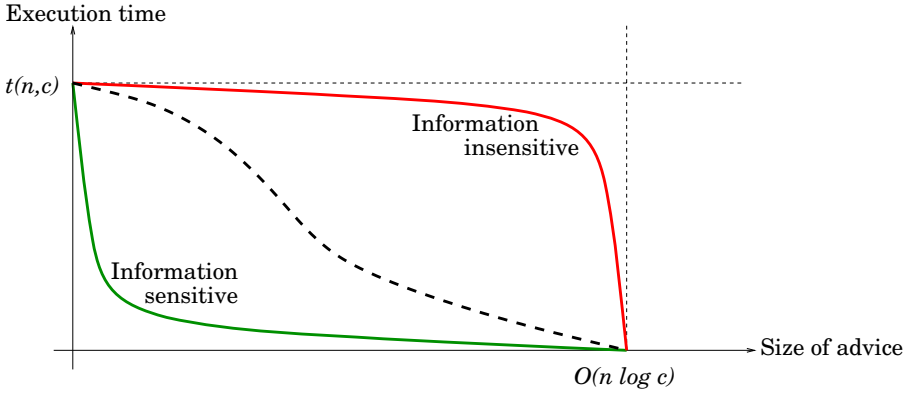
\*\* Andrzej Pelc was supported in part by NSERC discovery grant and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.



the instance of the considered problem. The traditional approach is actually *qualitative* in the sense that algorithms or impossibility results are proved under the assumption that the nodes are aware of specific parameters, e.g., the size of the network. It was proved that the impact of knowledge concerning the environment is significant in many areas of distributed computing, as witnessed by [8,18] where a lot of impossibility results and lower bounds are surveyed, many of them depending on whether or not the nodes are provided with partial knowledge of the topology of the network. A *quantitative* approach was recently introduced in [9], in which limitations of local computation can be estimated by establishing tradeoffs between the efficiency of the computation (number of steps, number of messages, etc.) and the amount of information provided to the nodes about their environment, independently of what kind of information they receive.

More precisely, we consider network computing with advice in the following context. A network is modeled as an undirected graph, where links represent communication channels between nodes. Nodes of  $n$ -node networks have distinct IDs from  $\{1, \dots, n\}$ , and communication ports at a node of degree  $d$  are labeled by distinct integers from  $\{1, \dots, d\}$ . A priori, every node knows only its own ID, and the labels of its ports. All additional knowledge available to the nodes of the graph (in particular knowledge concerning the topology and the labels of the rest of the graph), is modeled by an *oracle* providing *advice*. An oracle is a function  $\mathcal{O}$  whose arguments are networks, and the value  $\mathcal{O}(G)$ , for a network  $G = (V, E)$ , called the advice provided by the oracle to this graph, is in turn a function  $f : V \rightarrow \{0, 1\}^*$  assigning a finite binary string to every node  $v$  of the graph. Intuitively, the oracle looks at the entire labeled graph with IDs, and assigns to every node some information, coded as a string of bits. A node  $v$  is *informed* by the oracle if the string  $f(v)$  is non-empty. The *size* of the advice given by the oracle to a given graph  $G$  is the sum of the lengths of all strings it assigns to nodes. Hence this size is a measure of the amount of information about the graph, available to its nodes. Clearly, the size of advice is not smaller than the number of informed nodes. The objective is to establish tradeoffs between the size of the advice and the computational efficiency of the network.

Specifically, we focus on the distributed graph coloring problem, one of the most challenging problems in network computing for its practical applications, e.g., in radio networks [19], and for its relation with many other problems such as maximal independent set (MIS) [14,22] and symmetry breaking [11]. Initially, each node knows its ID from  $\{1, \dots, n\}$ . The  $c$ -coloring problem requires each node to compute a color in  $\{1, \dots, c\}$ , under the constraint that any two adjacent nodes have different colors. Computation proceeds in rounds following Linial's model defined in [16] (a.k.a., *LOCAL* model [24]). At each round, a node sends a message to each of its neighbors, receives messages from each of its neighbors, and performs some local computations. The *LOCAL* model does not put any limit on the message size and any restrictions on local computations because it is designed to estimate limitations of local computing. The complexity of  $c$ -coloring a graph  $G$  is measured by the number of rounds required to compute a proper



**Fig. 1.** Tradeoff between the execution time and the size of advice

$c$ -coloring. There is an obvious relation between the complexity of  $c$ -coloring and the maximum distance between two nodes that exchange information during the computation.

Coloring graphs using advice provided by oracle  $\mathcal{O}$  consists in designing an algorithm that is *unaware* of the graph  $G$  at hand but colors it distributively, as long as every node  $v$  of the graph  $G$  is provided with the string of bits  $f(v)$ , where  $f = \mathcal{O}(G)$ . Trivially, an advice of size  $O(n \log c)$  bits that provides the appropriate color to each node yields a coloring algorithm working in 0 rounds. On the other hand, an advice of size 0, i.e., providing no information, yields an algorithm running in  $t(n, c)$  rounds where  $t(n, c)$  is the complexity of the coloring problem in the usual distributed setting (i.e., with no advice).

The theory of network computing with advice allows us to establish tradeoffs between these two extreme cases. Different forms of tradeoffs are illustrated in Figure 1. This figure plots the execution time as a function of the size of advice (i.e., the amount of information given to the nodes). The execution time decreases as the size of advice increases, for instance such as illustrated by the dashed curve. Depending on how quickly the time decreases enables to roughly classify problems as "sensitive" or "insensitive" to information. A problem is *information sensitive* if few bits of information given to the nodes enable to decrease drastically the execution time. Conversely, a problem is *information insensitive* if the oracle must give a lot of information to the nodes for the execution time to decrease significantly. In this paper, we study the information sensitivity of graph coloring.

## 1.1 Our Results

To study the information sensitivity of graph coloring, we focus on lower bounds on the size of advice necessary for fast distributed coloring of cycles and trees, two important cases analyzed in depth by Linial in his seminal paper [16] (cf. also [10]).

We show that coloring a cycle is information insensitive. Precisely, we show that, for any constant  $k$ ,  $\Omega(n/\log^{(k)} n)$  bits of advice are needed in order to beat the  $\Theta(\log^* n)$  time of 3-coloring a cycle, where  $\log^{(k)} n$  denotes  $k$  iterations of  $\log n$ . This shows a huge gap between 3-coloring in time  $\Theta(\log^* n)$  and 3-coloring below this time: while the first can be done without any advice [6], the second requires almost as much information as if colors were explicitly assigned to nodes (which would take  $O(n)$  bits).

The result for cycles easily extends to oriented trees (i.e., rooted trees in which every node in the tree knows its parent in the tree), proving that, for any constant  $k$ ,  $\Omega(n/\log^{(k)} n)$  bits of advice are needed in order to beat the  $O(\log^* n)$  time of 3-coloring an oriented tree [10]. Coloring an oriented tree is thus also information insensitive.

The power of orienting a tree (i.e., giving an orientation of its edges toward a root), from the point of view of distributed coloring, was known since Linial [16] proved that no algorithm can color the  $d$ -regular unoriented tree of radius  $r$  in time at most  $\frac{2}{3}r$  by fewer than  $\frac{1}{2}\sqrt{d}$  colors. Hence 3-coloring unoriented trees essentially requires  $\Theta(D)$  rounds, where  $D$  is the diameter of the tree. Therefore, informing every node of the port leading to its parent in the tree results in decreasing the time of 3-coloring from  $\Omega(D)$  to  $O(\log^* n)$ . We revisit this result using our quantitative approach. Precisely, we aim at computing the amount of advice required to reach the  $O(\log^* n)$  time bound. It is known that  $O(n \log \log n)$  bits of advice enable to orient a tree (i.e., to select a root, and to give to every node the port number of the edge leading to its parent) with an algorithm working in 0 rounds [5], and  $O(n)$  bits of advice enable to orient a tree with an algorithm working in 1 round [4]. However, 3-coloring a tree in time  $\Theta(\log^* n)$  does not necessarily require to orient the tree. Nevertheless, we show that, for any constant  $k$ ,  $\Omega(n/\log^{(k)} n)$  bits of advice are needed in order to 3-color all  $n$ -node unoriented trees in time  $\Theta(\log^* n)$ . Thus, while for oriented trees 3-coloring in time  $O(\log^* n)$  can be done without any additional information [10], achieving the same efficiency for arbitrary trees requires almost as much information as if colors were explicitly assigned to nodes.

Finally, both for cycles and trees, even if oriented, we also show that  $\Omega(n)$  bits of advice are needed for 3-coloring in constant time (i.e., for 3-coloring to become a locally solvable problem). Thus constant-time coloring requires essentially as much information as if colors were explicitly assigned to nodes. In fact, our lower bounds hold not only for the total number of bits of advice given to nodes but also for the number of nodes that must be informed (i.e., the number of nodes that are given at least one bit of advice).

Although we formulate our results for the task of 3-coloring, they remain true for coloring with any constant number of colors, by slight technical modification of the proofs.

While our lower bound proofs present different technical challenges in the case of the cycle and that of trees, the underlying idea is similar in both cases. Linial [16] constructed the *neighborhood graph*  $N[G]$  of a graph  $G$  in order to estimate the time of coloring  $G$  using the chromatic number of  $N[G]$ . Since in our case

there is an oracle giving advice to nodes, we have to use a more complex tool in the lower bound argument. We also argue about the chromatic number of a suitably chosen graph  $H$  in order to bound coloring time of  $G$ . However, in our case, this graph depends on the oracle as well as on the time of coloring, and on the graph  $G$ , and hence it is very irregularly structured. We show that, if the number of nodes of  $G$  informed by the oracle is not too large, then  $H$  has a large chromatic number, and thus forces large coloring time of  $G$ . (Equivalently, if  $G$  can be colored fast then the advice must be large.) The main difficulty in our argument is to show the existence of a regularly structured subgraph (whose chromatic number can be bounded from below) in the highly irregularly structured graph  $H$ .

## 1.2 Related Work

Because of the intrinsic difficulty of computing the chromatic number of a graph in the sequential setting [12], or even to approximate it [3,7], the distributed computing literature dealing with graph coloring mostly focuses on the  $(\Delta + 1)$ -coloring problem, where  $\Delta$  denotes the maximum degree of the graph. In fact, the interest expressed for the  $(\Delta + 1)$ -coloring problem is also due to its intriguing relation with the maximal independent set (MIS) problem, already underlined by Linial in [16]. In particular, combining the best known algorithms for MIS [1,17] with the reduction from  $(\Delta + 1)$ -coloring to MIS by Linial yields a randomized  $(\Delta + 1)$ -coloring algorithm working in expected time  $O(\log n)$ . Using techniques described in [2] and [23], one can compute a  $(\Delta + 1)$ -coloring (as well as a MIS) of arbitrary graphs in deterministic time  $O(n^{1/\sqrt{\log n}})$ . For graphs of maximum degree bounded by  $\Delta$ ,  $(\Delta + 1)$ -coloring can be achieved in time  $O(\Delta \log n)$  (see [2]). [6] described a PRAM algorithm that can be easily transformed into an algorithm working in the  $\mathcal{LOCAL}$  model, computing a 3-coloring of oriented cycles in  $O(\log^* n)$  rounds. This bound is tight as proved by Linial [16]. Similarly, [10] described a 3-coloring of oriented trees working in  $O(\log^* n)$  rounds. The  $O(\Delta^2)$ -coloring algorithm in [16], working in  $O(\log^* n)$  rounds, can be easily converted into a  $(\Delta + 1)$ -coloring algorithm working in  $O(\Delta^2 + \log^* n)$  rounds, reaching the same complexity as the algorithm in [11]. [15] analyses what can be achieved in one round, and proves that no algorithm based on iterations of the application of a 1-round algorithm can achieve  $O(\Delta)$ -coloring in less than  $\Omega(\Delta/\log^2 \Delta + \log^* n)$  rounds. On the other hand, [15] presents a  $(\Delta + 1)$ -coloring algorithm working in  $O(\Delta \log \Delta + \log^* n)$  rounds, thus improving [2,11,16]. Recently, the power of orienting the network was also demonstrated in terms of bit complexity in [13].

## 2 Coloring Cycles with Advice

In order to prove the lower bounds listed in Section 1.1 on the size of advice needed for fast 3-coloring of all cycles, we prove the following result.

**Theorem 1.** *Suppose that an oracle  $\mathcal{O}$  informs at most  $q$  nodes in any  $n$ -node cycle. Then the time of 3-coloring of  $n$ -node cycles using oracle  $\mathcal{O}$  is  $\Omega(\log^*(n/q))$ . This result holds even if the cycle is oriented, i.e., even if the nodes have a consistent notion of clockwise and counterclockwise directions.*

*Proof.* Recall the definition of the directed graph  $B_{t,n}$  from [16]. Let  $s = 2t + 1 < n - 1$ . The nodes of the graph are sequences of length  $s$  of distinct integers from  $\{1, \dots, n\}$ . Intuitively, node  $(x_1, x_2, \dots, x_s)$  of the graph  $B_{t,n}$  represents the information acquired in time  $t$  by node  $x_{t+1}$  of a labeled directed cycle containing a segment  $(x_1, x_2, \dots, x_s)$ . Out-neighbors of node  $(x_1, x_2, \dots, x_s)$  are all nodes  $(x_2, x_3, \dots, x_s, y)$ , where  $y \neq x_1$ . Note that the chromatic number  $\chi(B_{t,n})$  is a lower bound on the number of colors with which an  $n$ -node cycle may be colored distributively in time  $t$ . Thus, by restricting attention to 3-coloring algorithms, this yields a lower bound on the time of 3-coloring.

It was proved in [16] that  $\chi(B_{t,n}) \geq \log^{(2t)} n$ . For any set  $X \subseteq \{1, \dots, n\}$  of size  $> s + 1$ , define  $B_{t,n}(X)$  to be the subgraph of  $B_{t,n}$  induced by all nodes  $(x_1, x_2, \dots, x_s)$  with  $x_i \in X$ , for all  $1 \leq i \leq s$ . It is easy to see that the graph  $B_{t,n}(X)$  is isomorphic to  $B_{t,|X|}$ .

Fix an oracle  $\mathcal{O}$  giving advice to all cycles of length  $n$ . Let  $q$  be the maximum number of nodes informed by oracle  $\mathcal{O}$  in any of these cycles. Without loss of generality we may assume that the number of bits given to any node is not more than needed to code all directed labeled cycles of length  $n$ , i.e.,  $\lceil \log(n-1)! \rceil$ . Consider a 3-coloring algorithm for cycles of length  $n$  using oracle  $\mathcal{O}$  and running in time  $t$ . If  $t \geq n/(2q) - 1$ , we are done. Hence suppose that  $t < n/(2q) - 1$  which implies  $s < n/q$ . We define the directed graph  $B_{t,n,\mathcal{O}}$  that will be crucial in our argument. The nodes of the graph are sequences  $((x_1, \alpha_1), (x_2, \alpha_2), \dots, (x_s, \alpha_s))$ , where  $x_i$  are distinct integers from  $\{1, \dots, n\}$  and  $\alpha_i$  are binary strings of length at most  $\lceil \log(n-1)! \rceil$ . Intuitively, node  $((x_1, \alpha_1), (x_2, \alpha_2), \dots, (x_s, \alpha_s))$  represents the total information acquired in time  $t$  by node  $x_{t+1}$  of a labeled directed cycle containing a segment  $(x_1, x_2, \dots, x_s)$ , including labels of nodes at distance at most  $t$  and advice given to them by the oracle. There exists a (directed) edge from node  $v = ((x_1, \alpha_1), (x_2, \alpha_2), \dots, (x_s, \alpha_s))$  to a node  $w = ((x_2, \alpha_2), \dots, (x_s, \alpha_s), (y, \beta))$  and if there exists a labeled directed cycle of length  $n$  containing the segment  $(x_1, x_2, \dots, x_s, y)$ , such that oracle  $\mathcal{O}$  applied to this cycle gives advice  $\alpha_1, \alpha_2, \dots, \alpha_s, \beta$  to nodes  $x_1, x_2, \dots, x_s, y$ , respectively. We will say that the segment  $(x_1, x_2, \dots, x_s, y)$  of such a cycle *induces* this directed edge. Similarly as above, the chromatic number  $\chi(B_{t,n,\mathcal{O}})$  is a lower bound on the number of colors with which the cycle may be colored distributively in time  $t$ , using oracle  $\mathcal{O}$ . Note that a coloring algorithm using oracle  $\mathcal{O}$  does not need to assign a color to all nodes  $((x_1, \alpha_1), (x_2, \alpha_2), \dots, (x_s, \alpha_s))$  of  $B_{t,n,\mathcal{O}}$ . Indeed, it is possible that there is no cycle containing the segment  $(x_1, x_2, \dots, x_s)$ , such that oracle  $\mathcal{O}$  applied to this cycle gives advice  $\alpha_1, \alpha_2, \dots, \alpha_s$  to nodes  $x_1, x_2, \dots, x_s$ , respectively. However, by definition, such “non-legitimate” nodes are isolated in the graph  $B_{t,n,\mathcal{O}}$  and hence they do not affect its chromatic number.

We will establish a lower bound on the chromatic number of  $B_{t,n,\mathcal{O}}$ , and then show how to deduce from it a lower bound on the time of 3-coloring with oracle  $\mathcal{O}$ .

To this end it is sufficient to focus on the subgraph  $\tilde{B}_{t,n,\mathcal{O}}$  of  $B_{t,n,\mathcal{O}}$  induced by the nodes  $((x_1, \alpha_1), (x_2, \alpha_2), \dots, (x_s, \alpha_s))$ , with all  $\alpha_i$  being empty strings. By definition, the graph  $\tilde{B}_{t,n,\mathcal{O}}$  is isomorphic to a subgraph of  $B_{t,n}$  and has the same number of nodes as  $B_{t,n}$ . By a slight abuse of notation we will identify  $\tilde{B}_{t,n,\mathcal{O}}$  with this subgraph of  $B_{t,n}$ .

*Claim 1.* For  $n/q$  sufficiently large, there exists a set  $X$  of size

$$k = \left\lfloor \left( \frac{n}{q(s+1)} \right)^{1/(s+1)} \right\rfloor$$

such that  $B_{t,n}(X)$  is a subgraph of  $\tilde{B}_{t,n,\mathcal{O}}$ .

Due to lack of space, the proof of Claim 1 is omitted.

In view of Claim 1, the chromatic number of  $B_{t,n,\mathcal{O}}$  can be bounded as follows (for  $n/q$  sufficiently large):

$$\chi(B_{t,n,\mathcal{O}}) \geq \log^{(s-1)} k = \log^{(s-1)} \left( \frac{n}{q(s+1)} \right)^{1/(s+1)}.$$

Since  $t$  is the running time of a 3-coloring algorithm for cycles of length  $n$  using oracle  $\mathcal{O}$ , we have  $\chi(B_{t,n,\mathcal{O}}) \leq 3$ , which implies  $\log^{(s-1)} \left( \frac{n}{q(s+1)} \right)^{1/(s+1)} \leq 3$ . In order to finish the argument, it is enough to prove that  $s \geq \frac{1}{5} \log^*(n/q)$ . Suppose not. Thus  $n/q \geq 2^{2^{16}}$ . For such large  $n/q$  we have

$$\log \frac{n}{q(s+1)} > \log \frac{n}{q} - \log \log^* \frac{n}{q} \geq \frac{1}{2} \log \frac{n}{q}.$$

Hence

$$\frac{1}{s+1} \log \frac{n}{q(s+1)} > \frac{1}{2(s+1)} \log \frac{n}{q} \geq \frac{1}{2 \log^* \frac{n}{q}} \log \frac{n}{q} \geq \log \log \frac{n}{q}.$$

This implies

$$\left( \frac{n}{q(s+1)} \right)^{1/(s+1)} > \log \frac{n}{q},$$

and

$$3 \geq \log^{(s-1)} \left( \frac{n}{q(s+1)} \right)^{1/(s+1)} > \log^{(s)} \frac{n}{q}.$$

Thus  $s \geq \log^* \frac{n}{q} - 2$ , which contradicts the assumption  $s < \frac{1}{5} \log^*(n/q)$ .  $\square$

Theorem [1](#) has several interesting consequences. The following corollary proves that transforming the 3-coloring problem into a locally solvable problem (in the sense of [\[20\]](#)) essentially requires to give the solution to the nodes.

**Corollary 1.** *Any distributed algorithm that produces a 3-coloring of all cycles of length  $n$  in constant time requires advice for  $\Omega(n)$  nodes.*

The next corollary proves that 3-coloring of cycles is information insensitive.

**Corollary 2.** *Any distributed algorithm that produces a 3-coloring of all cycles of length  $n$  in time  $o(\log^* n)$  requires advice for  $\Omega(n/\log^{(k)} n)$  nodes, for any constant  $k$ .*

### 3 Coloring Trees with Advice

Theorem 1 concerning cycles has an interesting consequence concerning trees, that proves that 3-coloring is information insensitive in *oriented* trees. Recall that a tree is oriented if it is rooted, and every node is aware of which of its incident edges leads to its parent in the tree. If there exists an oracle  $\mathcal{O}$  informing at most  $q$  nodes in any  $n$ -node oriented tree, and a 3-coloring algorithm  $\mathcal{A}$  using  $\mathcal{O}$  and working in  $t(n)$  rounds, then there exists an oracle  $\mathcal{O}'$  informing at most  $q + 2$  nodes in any  $n$ -node oriented cycle, and a 3-coloring algorithm  $\mathcal{A}'$  using  $\mathcal{O}'$  and working in  $t(n) + 1$  rounds.  $\mathcal{O}'$  picks arbitrarily two neighboring nodes  $x$  and  $y$  in the cycle. Assume that  $y$  is the neighbor of  $x$  in the counterclockwise direction.  $\mathcal{O}'$  gives the advice (tail) to  $x$ , and the advice  $(t(n), \text{root})$  to  $y$ . The  $i$ th node  $v$  in the cycle, counting counterclockwise from  $x$ , receives from  $\mathcal{O}'$  the advice  $f(v_i)$  given by  $\mathcal{O}$  to the node  $v_i$  at distance  $i$  from the root of the oriented path  $P$  rooted at one of its two extremities, where  $f = \mathcal{O}(P)$ .  $\mathcal{A}'$  proceeds in  $t(n) + 1$  rounds. During rounds 1 to  $t(n)$ ,  $\mathcal{A}'$  simply executes  $\mathcal{A}$ , for which nodes  $x$  and  $y$  just act as if they would be respectively the tail and the root of a directed path from  $x$  to  $y$ . At round  $t(n) + 1$  of  $\mathcal{A}'$ , the root node  $y$  checks if its color is different from  $x$ . If not, it takes a color distinct from the colors of its two neighbors. This simple reduction enables to establish the following corollary of Theorem 1 proving that 3-coloring oriented trees is information insensitive.

**Corollary 3.** *Suppose that an oracle  $\mathcal{O}$  informs at most  $q$  nodes in any  $n$ -node oriented tree. Then the time of 3-coloring of  $n$ -node oriented trees using oracle  $\mathcal{O}$  is  $\Omega(\log^*(n/q))$ . Thus in particular any distributed algorithm that produces a 3-coloring of all  $n$ -node oriented trees in time  $o(\log^* n)$  requires advice for  $\Omega(n/\log^{(k)} n)$  nodes, for any constant  $k$ .*

The main result of this section is a lower bound on the size of advice necessary for fast coloring of all  $n$ -node *unoriented* trees. In fact we will show that this bound holds already for the class of all unoriented complete  $d$ -regular trees. These are trees  $T_{d,r}$  such that each leaf is at distance  $r$  from the root, and each internal node has degree  $d$ . It should be stressed that the notion of root and children is brought up only to facilitate the definition. From the point of view of nodes, the tree is not rooted (a node does not have information which neighbor is its parent).

**Theorem 2.** *Fix  $d \geq 37$ . Any 3-coloring algorithm working in time  $t$  for the class of  $n$ -node unoriented complete  $d$ -regular trees requires advice for at least  $\frac{n}{d^{2t}}$  nodes.*

*Proof.* Fix  $d \geq 37$ ,  $t > 0$ , and  $r > 2t + 3$ . Consider any node  $v$  of the tree  $T_{d,r}$  at distance at least  $t + 1$  from all leaves. The number of nodes at distance at most  $t$

from  $v$  will be denoted by  $\alpha(t)$ . We have  $\alpha(t) = d \cdot \sum_{i=0}^{t-1} (d-1)^i \leq 2(d-1)^t - 1$ . Consider an edge  $e$  of the tree  $T_{d,r}$  whose both extremities are at distance at least  $t+1$  from all leaves. The subtree induced by the set of nodes at distance at most  $t$  from one of these extremities will be called the *bow-tie* of  $T_{d,r}$  based on edge  $e$ . The number of nodes in this bow-tie will be denoted by  $\beta(t)$ . We have  $\beta(t) = \alpha(t) + 1 + (d-1)^t \leq 3(d-1)^t$ .

Consider the tree  $T_{d,r}$  with a labeling  $\Phi$  of nodes and ports.  $\Phi$  labels all nodes by distinct integers from  $\{1, \dots, n\}$ , where  $n = 1 + \alpha(r)$ , and labels all ports at internal nodes by integers from  $\{1, \dots, d\}$ . For any such labeled tree, consider its subtrees of the form  $N(v, t, \Phi)$ , where  $t$  is a positive integer and  $v$  is a node of  $T_{d,r}$  at distance at least  $t+1$  from any leaf of  $T_{d,r}$ .  $N(v, t, \Phi)$  is defined as the labeled subtree of  $T_{d,r}$  induced by all nodes at distance at most  $t$  from  $v$ . Note that if restrictions of labelings  $\Phi$  and  $\Phi'$  to the subtree of  $T_{d,r}$  induced by all nodes at distance at most  $t$  from  $v$  are identical, then  $N(v, t, \Phi) = N(v, t, \Phi')$ . Consider the following graph  $G_t(T_{d,r})$ . The nodes of the graph are all subtrees  $N(v, t, \Phi)$  of  $T_{d,r}$  for all possible nodes  $v$  and labelings  $\Phi$  of nodes and ports of  $T_{d,r}$ . Two nodes of  $G_t(T_{d,r})$  are adjacent, if and only if, they are of the form  $N(v, t, \Phi)$  and  $N(v', t, \Phi)$ , for some labeling  $\Phi$ , with  $v$  and  $v'$  adjacent in  $T_{d,r}$ . Note that the graph  $G_t(T_{d,r})$  is a subgraph of the  $t$ -neighborhood graph of  $T_{d,r}$ , defined in [16]. Moreover, it follows from [16] that the chromatic number  $\chi(G_t(T_{d,r}))$  is a lower bound on the number of colors with which the tree  $T_{d,r}$  may be colored distributively in time  $t$ , and that  $\chi(G_t(T_{d,r})) \geq \frac{1}{2}\sqrt{d}$ , if  $t < 2r/3$ . Also, for any set  $X \subseteq \{1, \dots, n\}$ , we define the graph  $G(X)$  as the subgraph of  $G_t(T_{d,r})$  induced by nodes with labels from the set  $X$ . Note that, for  $|X| = 1 + \alpha(s)$ , for some positive integer  $s \leq r$ , the graph  $G(X)$  is isomorphic to  $G_t(T_{d,s})$ .

Fix an oracle  $\mathcal{O}$  giving advice to all  $n$ -node labeled trees  $T_{d,r}$ . Let  $q$  be the maximum number of nodes informed by oracle  $\mathcal{O}$  in any of these trees. Without loss of generality we may assume that the number of bits given to any node is not more than needed to code all  $n$ -node labeled trees  $T_{d,r}$ . There are  $d^{\alpha(r-1)}$  port labelings of  $T_{d,r}$ , and for each such labeling there are  $n!$  ways to label nodes. Hence the number of bits needed to code these trees is at most  $\lceil \log(d^{\alpha(r-1)}n!) \rceil$ . Consider a 3-coloring algorithm for  $n$ -node labeled trees  $T_{d,r}$  using oracle  $\mathcal{O}$  and running in time  $t$ . We define the following graph  $G_{t,\mathcal{O}}(T_{d,r})$ . Nodes of this graph are couples of the form  $(N(v, t, \Phi), f)$ , where  $N(v, t, \Phi)$  is the tree defined above and  $f$  is a function from nodes of this tree into the set of binary strings of length at most  $\lceil \log(d^{\alpha(r-1)}n!) \rceil$ . Intuitively, the value  $f(w)$  is the advice given to node  $w$  of  $N(v, t, \Phi)$  by the oracle, and the entire couple  $(N(v, t, \Phi), f)$  represents the total information acquired in time  $t$  by node  $v$ , including advice given to nodes of  $N(v, t, \Phi)$  by the oracle. Edges of the graph  $G_{t,\mathcal{O}}(T_{d,r})$  are defined as follows. There is an (undirected) edge between two nodes of  $G_{t,\mathcal{O}}(T_{d,r})$  if these nodes are of the form  $(N(v, t, \Phi), f)$  and  $(N(v', t, \Phi), f')$ , for some labeling  $\Phi$ , where  $v$  and  $v'$  are adjacent in  $T_{d,r}$  and for all nodes  $w$  of  $N(v, t, \Phi)$  and  $w'$  of  $N(v', t, \Phi)$ , the values  $f(w)$  and  $f'(w')$  are advice strings given to nodes  $w$  and  $w'$ , respectively, by oracle  $\mathcal{O}$  for the tree  $T_{d,r}$  labeled by  $\Phi$ . We will say that this edge of  $G_{t,\mathcal{O}}(T_{d,r})$  is induced by the bow-tie based on edge  $\{v, v'\}$  of the tree  $T_{d,r}$  labeled by  $\Phi$ .



The chromatic number  $\chi(G_{t,\mathcal{O}}(T_{d,r}))$  is a lower bound on the number of colors with which the tree  $T_{d,r}$  may be colored distributively in time  $t$ , using oracle  $\mathcal{O}$ . Similarly as in the case of the cycle, there may be “non-legitimate” nodes in  $G_{t,\mathcal{O}}(T_{d,r})$  but they are isolated and thus do not affect the chromatic number.

In order to establish a lower bound on the chromatic number of  $G_{t,\mathcal{O}}(T_{d,r})$ , it is sufficient to focus on the subgraph  $\tilde{G}_{t,\mathcal{O}}(T_{d,r})$  induced by the nodes

$$(N(v, t, \Phi), f)$$

with  $f$  being a function giving the empty string to all nodes. By definition, the graph  $\tilde{G}_{t,\mathcal{O}}(T_{d,r})$  is isomorphic to a subgraph of  $G_t(T_{d,r})$  and has the same number of nodes as  $G_t(T_{d,r})$ . Similarly as before we will identify  $\tilde{G}_{t,\mathcal{O}}(T_{d,r})$  with this subgraph of  $G_t(T_{d,r})$ .

*Claim 2.* Let  $\nu(k)$  be the number of sets  $X$  of size  $k$ , such that the graph  $G(X)$  is not a subgraph of  $\tilde{G}_{t,\mathcal{O}}(T_{d,r})$ . Then

$$\nu(k) \leq \frac{2 \cdot q \cdot n! \cdot d^{4d^t}}{n \cdot (n - \beta(t))!} \cdot \binom{n - \beta(t)}{k - \beta(t)}.$$

Due to lack of space, the proof of Claim 2 is omitted.

Suppose that  $\nu(k) < \binom{n}{k}$ . Then there exists a set  $X$  of size  $k$  for which  $G(X)$  is a subgraph of  $\tilde{G}_{t,\mathcal{O}}(T_{d,r})$ . Since  $k = \alpha(s)$  for  $s > 3t/2$ , it follows from [16] that the chromatic number of the graph  $G(X)$  (and thus also of the graph  $G_{t,\mathcal{O}}(T_{d,r})$ ) is at least  $\frac{1}{2}\sqrt{d}$ , which is larger than 3 for  $d \geq 37$ . This contradicts the fact that we consider a 3-coloring algorithm running in time  $t$ . Hence we may assume  $\nu(k) \geq \binom{n}{k}$ . From Claim 2, this implies

$$\frac{2 \cdot q \cdot n! \cdot d^{4d^t}}{n \cdot (n - \beta(t))!} \cdot \binom{n - \beta(t)}{k - \beta(t)} \geq \binom{n}{k}$$

and hence the number  $q$  of informed nodes satisfies

$$q \geq \frac{n \cdot (k - \beta(t))!}{2 \cdot d^{4d^t} \cdot k!} \geq \frac{n}{2 \cdot d^{4d^t} \cdot k^{\beta(t)}}.$$

Since  $k = \alpha(\lfloor \frac{3}{2}t + 1 \rfloor) \leq d^{\frac{7}{2}t}$  and  $\beta(t) \leq 3(d - 1)^t$ , we have

$$q \geq \frac{n}{2 \cdot d^{4d^t} \cdot d^{\frac{7}{2}t \cdot 3(d-1)^t}} \geq \frac{n}{d^{d^{2t}}}.$$

□

**Remark.** By considering trees of a sufficiently large constant degree (instead of just degree  $d \geq 37$ ) we can generalize the above result to the case of  $c$ -coloring, for any constant  $c$ .

Theorem 2 has several interesting consequences. The following corollary proves that lack of cycles does not help in coloring a network since transforming the 3-coloring problem in trees into a locally solvable problem essentially requires, as for cycles, to give the solution to the nodes.

**Corollary 4.** *Any distributed algorithm that produces a 3-coloring of all  $n$ -node trees in constant time requires advice for  $\Omega(n)$  nodes.*

The next corollary proves that reaching the  $O(\log^* n)$  bound in unoriented trees requires lot of advice. This should be contrasted with the fact that  $O(\log^* n)$  is the complexity of 3-coloring of *oriented* trees, without advice.

**Corollary 5.** *Any distributed algorithm that produces a 3-coloring of all  $n$ -node unoriented trees in time  $O(\log^* n)$  requires advice for  $\Omega(n/\log^{(k)} n)$  nodes, for any constant  $k$ .*

## 4 Conclusion

We presented lower bounds on the amount of advice that has to be given to nodes of cycles and of trees in order to produce distributively a fast 3-coloring of these networks. Although our lower bounds are very close to the obvious upper bound  $O(n)$ , some interesting detailed questions concerning the trade-offs between the size of advice and the time of coloring remain open, even for cycles and trees. In particular, what is the minimum number of bits of advice to produce a 3-coloring of every  $n$ -node cycle or tree in a given time  $t = o(\log^* n)$ ? More generally, what is the information sensitivity of coloring arbitrary graphs? For arbitrary graphs, it is natural to consider the maximum degree  $\Delta$  as a parameter, and seek distributed  $(\Delta+1)$ -coloring. It was proved in [15] that a  $(\Delta+1)$ -coloring can be produced in time  $O(\Delta \log \Delta + \log^* n)$ . What is the minimum number of bits of advice to produce a  $(\Delta+1)$ -coloring in time  $O(\log^* n)$ ? And in constant time? We conjecture that for the former task  $O(n)$  bits of advice are sufficient, and for the latter  $\Omega(n \log \Delta)$  bits of advice are needed.

## References

1. Alon, N., Babai, L., Itai, A.: A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *J. Algorithms* 7(4), 567–583 (1986)
2. Awerbuch, B., Goldberg, A., Luby, M., Plotkin, S.: Network Decomposition and Locality in Distributed Computation. In: 30th Symp. on Foundations of Computer Science(FOCS), pp. 364–369 (1989)
3. Bellare, M., Goldreich, O., Sudan, M.: Free Bits, PCPs, and Nonapproximability – Towards Tight Results. *SIAM Journal on Computing* 27(3), 804–915 (1998)
4. Cohen, R., Fraigniaud, P., Ilcinkas, D., Korman, A., Peleg, D.: Label-Guided Graph Exploration by a Finite Automaton. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 335–346. Springer, Heidelberg (2005)
5. Cohen, R., Fraigniaud, P., Ilcinkas, D., Korman, A., Peleg, D.: Labeling Schemes for Tree Representation. In: Pal, A., Kshemkalyani, A.D., Kumar, R., Gupta, A. (eds.) IWDC 2005. LNCS, vol. 3741, pp. 13–24. Springer, Heidelberg (2005)
6. Cole, R., Vishkin, U.: Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms. In: 18th ACM Symp. on Theory of Computing (STOC), pp. 206–219. ACM Press, New York (1986)

7. Feige, U., Kilian, J.: Zero Knowledge and the Chromatic Number. *J. Comput. Syst. Sci.* 57(2), 187–199 (1998)
8. Fich, F., Ruppert, E.: Hundreds of impossibility results for distributed computing. *Distributed Computing* 16, 121–163 (2003)
9. Fraigniaud, P., Ilcinkas, D., Pelc, A.: Oracle size: a new measure of difficulty for communication tasks. In: 25th ACM Symp. on Principles of Distributed Computing (PODC), pp. 179–187. ACM Press, New York (2006)
10. Goldberg, A., Plotkin, S.: Efficient parallel algorithms for  $(\Delta + 1)$ -coloring and maximal independent set problems. In: 19th ACM Symp. on Theory of Computing (STOC), pp. 315–324. ACM Press, New York (1987)
11. Goldberg, A., Plotkin, S., Shannon, G.: Parallel symmetry-breaking in sparse graphs. In: 19th ACM Symp. on Theory of Computing (STOC), pp. 315–324. ACM Press, New York (1987)
12. Karp, R.: Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*, 85–103 (1972)
13. Kothapalli, K., Onus, M., Scheideler, C., Schindelbauer, C.: Distributed coloring in  $O(\sqrt{\log n})$  bit rounds. In: 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE Computer Society Press, Los Alamitos (2006)
14. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What cannot be computed Locally! In: 23th ACM Symp. on Principles of Distributed Computing (PODC), pp. 300–309. ACM Press, New York (2004)
15. Kuhn, F., Wattenhofer, R.: On the complexity of distributed graph coloring. In: 25th ACM Symp. on Principles of Distributed Computing (PODC), pp. 7–15. ACM Press, New York (2006)
16. Linial, N.: Locality in distributed graph algorithms. *SIAM J. on Computing* 21(1), 193–201 (1992)
17. Luby, M.: A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.* 15(4), 1036–1053 (1986)
18. Lynch, N.: A hundred impossibility proofs for distributed computing. In: 8th ACM Symp. on Principles of Distributed Computing (PODC), pp. 1–28. ACM Press, New York (1989)
19. Moscibroda, T., Wattenhofer, R.: Coloring unstructured radio networks. In: 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA), pp. 39–48. ACM Press, New York (2005)
20. Naor, M., Stockmeyer, L.: What can be computed locally? In: 25th ACM Symposium on Theory of Computing (STOC), pp. 184–193. ACM Press, New York (1993)
21. Panconesi, A., Rizzi, R.: Some simple distributed algorithms for sparse networks. *Distributed Computing* 14, 97–100 (2001)
22. Panconesi, A., Srinivasan, A.: Improved distributed algorithms for coloring and network decomposition problems. In: 24th ACM Symp. on Theory of Computing (STOC), pp. 581–592. ACM Press, New York (1992)
23. Panconesi, A., Srinivasan, A.: On the complexity of distributed network decomposition. *Journal of Algorithms* 20(2), 356–374 (1996)
24. Peleg, D.: *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics (2000)

# Private Multiparty Sampling and Approximation of Vector Combinations

Yuval Ishai<sup>1</sup>, Tal Malkin<sup>2</sup>, Martin J. Strauss<sup>3</sup>, and Rebecca N. Wright<sup>4</sup>

<sup>1</sup> Computer Science Dept., Technion, Haifa 32000 Israel

<sup>2</sup> Dept. of Computer Science, Columbia University, New York, NY 10025 USA

<sup>3</sup> Depts. of Math and EECS, University of Michigan, Ann Arbor, MI 48109 USA

<sup>4</sup> Computer Science Dept., Stevens Institute of Technology, Hoboken, NJ 07030 USA

**Abstract.** We consider the problem of private efficient data mining of vertically-partitioned databases. Each of several parties holds a column of a data matrix (a vector) and the parties want to investigate the componentwise combination of their vectors. The parties want to minimize communication and local computation while guaranteeing privacy in the sense that no party learns more than necessary. Sublinear-communication private protocols have been primarily been studied only in the two-party case. We give efficient multiparty protocols for sampling a row of the data matrix and for computing arbitrary functions of a row, where the row index is additively shared among two or more parties. We also give protocols for approximating the componentwise sum, minimum, or maximum of the columns in which the communication and the number of public-key operations are at most polynomial in the size of the small approximation and polylogarithmic in the number of rows.

## 1 Introduction

There are many real-life scenarios in which several mutually distrusting entities (e.g., credit agencies, hospitals, or network carriers) have a common interest in obtaining useful summaries of their combined data. For instance, the parties may want to learn basic statistics on the combined data, measure the amount of similarity between their inputs, or detect irregularities or fraud by means of identifying major discrepancies in common entries.

In our setting, each of  $M$  parties  $P_1, \dots, P_M$  has a length- $N$  (column) vector, denoted  $x^m$  for  $P_m$  as a private input. For some  $M$ -ary function  $f$ , the parties want to compute a length- $N$  vector  $y$  whose  $n$ 'th component  $y_n$  is given by  $f(x_n^1, x_n^2, \dots, x_n^M)$ . We write this as  $y = f(\mathbf{x})$  and call  $y$  a *combination* of the parties' inputs. Examples of combination functions are the identity function (where an  $M$ -ary identity function simply returns its  $M$  inputs as outputs), or the sum function (that returns the sum of its  $M$  inputs).

If  $N$  is small, general secure multiparty computation [17,18] can be used efficiently. We provide solutions that are efficient even when  $N$  (and therefore  $y$ ) is very large. We aim for solutions with local computation at most polynomial in  $N$  and  $M$  and communication at most polynomial in  $M$  and

$\log(N)$ . Towards this end, we provide solutions in which the parties do not compute  $y$ , but rather some moderately sized “approximation” or “summary” of it. In the non-private setting, there is a rich body of work demonstrating that communication complexity can be dramatically improved by using an approximate solution that allows a small error probability (e.g., [11,22],[16,4,8]). Generally, however, these existing approximations are not private, meaning that some of the parties may learn more than what follows from their inputs and outputs.

Useful approximations that the parties may wish to compute include a sample of components in  $y$  or a known transform of  $y$  such as the Fourier transform; statistical summaries of  $y$ , such as a norm of  $y$ ; an approximate reconstruction of  $y$ , such as a piecewise-constant approximation with error nearly optimal according to some norm; and a succinct data structure to which one can pose queries about  $y$ . As a concrete class of examples, we focus on the quantity  $\|y\|_a = (\sum_n y_n^a)^{1/a}$ , which we call a *norm* of  $y$  and denote by  $\ell^a$ . (Technically, it is only a norm for certain values of  $a$ .) One can regard a norm of  $y$  as an approximation to the vector  $y$ . A useful special case is the problem of multiparty set intersection size. The parties have subsets  $A_1, A_2, \dots, A_M$  of a known universe of size  $N$  and we want to compute  $|\bigcap_m A_m|$ . (In this case the vector combination function  $f$  is the bitwise-AND, namely  $y = f(x^1, \dots, x^M) = \bigwedge_m x^m$ , and the output the parties seek is  $\|y\|_1 = \sum_n y_n$ .) Even in the two-party case and without any privacy requirements, it is impossible to achieve a constant multiplicative approximation with a sublinear amount of communication. We thus settle for an additive approximation, up to an error of  $\pm \epsilon N$ .

**Our Results.** We present communication-efficient solutions to a large class of useful special cases of efficient private distributed computation. Our results also have useful consequences for the general theory of secure multiparty computation with sublinear communication and highlight some qualitatively interesting differences between the two-party and the multiparty case.

Specifically, we show solutions to two multiparty problems: private multiparty sampling (Section 3) and private approximation of vector combinations (Section 4). Our private multiparty sampling solution uses two-party private information retrieval (PIR) as a building block. Private multiparty sampling itself is a useful tool in a wide range of private approximation scenarios, such as communication-efficient multiparty approximations of set intersection and of the  $\ell^2$ -norm of the sum of  $M$  input vectors. For private approximation of vector combinations, we consider approximations to the componentwise sum, minimum, or maximum over  $M$  vectors of integers. In a private computation setting, this problem is usually not interesting in the two-party case, as the input vector of one party together with the output vector allows her to determine most (if not all) of the other party’s input. However, when there is a larger number of parties, this problem becomes natural.

In the full version of the paper, we also discuss some interesting consequences of our results to the general problem of reducing sublinear-communication secure multiparty computation to two-party PIR.

**Related Work.** The approach of constructing secure sublinear-communication protocols was initiated in the context of private information retrieval [6] and further studied both in other specific contexts (e.g., [24]) and in more general settings [27]. Freedman et al. [13] give an efficient two-party protocol for approximating the size of an intersection of sets from a universe of size  $N$  with additive error small compared with  $N$ . That is, they compute the AND of  $x_n^1$  and  $x_n^2$  at some random position  $n$  unknown to the parties. Our results in Section 3 can be regarded as a generalization of this result to more than two parties and to functions other than the AND of bits. Indyk and Woodruff [20] give a two-party, polylog-communication private protocol for approximating the  $\ell^2$ -norm of the difference (or sum) of vector inputs. Our results of Section 3 can be used to extend their result to more than two parties.

Naor and Nissim [27] present a general compiler of any two-party protocol into a private protocol which preserves the communication, up to polynomial factors. This compiler, however, generally requires an exponential amount of local computation and thus it is not directly useful for the approximation problems we consider. Nevertheless, for the classes of functions for which their compilation technique efficiently applies, our results of Section 3 can be used to efficiently generalize their protocols from two parties to more than two parties providing security against any subset of the parties.

## 2 Background

### 2.1 Privacy

When mutually suspicious parties conduct a computation on their joint data, they want to guarantee that the privacy of their inputs is protected, in the sense that the protocol leaks nothing more than necessary about their inputs. In our context, where the computed output of the parties is an approximation  $g(y)$  of a combination vector  $y = f(\mathbf{x})$ , we consider two types of privacy guarantee.

**Privacy with respect to the output.** This is the traditional privacy guarantee for secure multiparty computation [5, 17] of output  $g(f(x))$  from inputs  $x$ . Given a functionality mapping the parties' inputs to (possibly randomized) outputs, the requirement is that no set of parties can learn anything about the inputs of other parties from protocol messages beyond their own inputs and outputs. In our setting, this privacy guarantee is the desired one in applications where the parties are willing to disclose the result  $g(f(x))$  but do not want to reveal any other information.

**Privacy with respect to the combination vector.** This kind of guarantee, introduced in [12], is “privacy of approximations.” A protocol is a private approximation protocol for  $f$  if its output is (with high probability) a good

approximation<sup>1</sup> for the exact output of  $f$ , and moreover each set of parties learn nothing additional from protocol messages (including the actual output of the protocol) beyond their own inputs and the ideal output,  $f(x)$ . Stated in our context, while the output of the protocol is  $g(f(x))$ , the privacy guarantee is that nothing is leaked that doesn't follow from  $f(x)$ . This is a weaker privacy guarantee (which admits much more efficient results in some cases). This is appropriate in applications where the parties do not mind disclosing  $f(x)$ , but do not want any further information leaked.

**Adversary model.** All of our protocols are computationally private against a non-adaptive, semi-honest (passive) adversary corrupting an arbitrary subset of the  $M$  parties. Naor and Nissim [27] showed how to upgrade security in the semi-honest model into security in the malicious model with a low communication overhead. Thus, from a theoretical point of view, our solutions can be modified to provide security in the malicious model while remaining communication-efficient. From a more practical point of view, most of our protocols provide reasonable security guarantees against malicious adversaries even without modification. In particular, the highly efficient protocols in Section 4 are fully private against a malicious adversary in the sense that it cannot learn more about the inputs of uncorrupted parties than is allowed in an ideal function evaluation.

## 2.2 PIR and Oblivious Transfer

We make use of private information retrieval (PIR) and oblivious transfer (OT). A PIR protocol allows a receiver to retrieve an item from a large database held by a sender without revealing which item she is after, and while using only a small amount of communication [6,23]. A *symmetric* PIR (SPIR) protocol [15,28], or sublinear-communication oblivious transfer [30,11], further guarantees that the receiver cannot learn more than a single entry of the database. Any PIR protocol can be used (in a black-box way) to obtain an OT protocol with similar communication complexity [28,9]. The communication complexity of PIR and OT on a database containing  $N$  short entries (say, bits) can be made as small as  $O(N^\epsilon)$  for an arbitrary constant  $\epsilon > 0$ , assuming that a homomorphic encryption scheme exists [23,31,26], or even polylogarithmic in  $N$  under more specific assumptions [3,25,14]. In the following, when referring to OT (and its variants) we always assume the communication to be sublinear in  $N$ .

## 3 Private Multiparty Sampling

In this section, we consider the challenge of extending a private sampling technique that lies in the core of previous two-party private approximation protocols [12,13,20], to the multiparty setting. Private multiparty sampling allows  $M$

<sup>1</sup> In this paper, we do not insist on a specific notion of approximation, such as additive or multiplicative one. Instead, we accept whatever function  $g$  the parties want to compute as a useful approximation of  $f$ , and focus on guaranteeing privacy of the protocol. For example, statistical summaries such as the norm of the vector are often used as an approximation of a vector.

parties, each holding a database  $x^m$ , to privately obtain  $f(x_r^1, \dots, x_r^M)$  where  $f$  is some fixed  $M$ -argument functionality (say, exclusive-or) and  $r$  is an index of a random entry that should remain secret.

When there is no restriction on communication complexity, a private multiparty sampling protocol can be constructed by making a black-box use of an arbitrary OT protocol, as follows from general techniques for secure multiparty computation [18,19,21]. Interestingly, such a construction becomes more challenging to obtain in the domain of sublinear-communication protocols. Further, this difficulty does not arise in the two-party setting and only seems to crop up when there are three or more parties. Indeed, a simple black-box construction of two-party private sampling from an arbitrary OT protocol (alternatively, PIR) is given in [12]. This construction maintains the communication complexity of the underlying OT protocol.<sup>2</sup> Thus, sublinear-communication OT (alternatively, PIR) can be used as a black box to realize sublinear-communication two-party private sampling. We do not know whether the same is true in the multiparty setting; this is an interesting question left open by our work.

Instead, we present a private multiparty sampling protocol that makes black-box use of PIR but it relies on the assumption that the underlying PIR protocol has only a single round of interaction (which is the case for almost all PIR protocols from the literature). The round complexity of our protocol is linear in the number of parties. In the full version of this paper, we also show how to implement private multiparty sampling with non-black-box use of an underlying PIR primitive. Although this is less efficient, it can be based on an arbitrary PIR protocol (even one using multiple rounds) and can yield a constant-round protocol (assuming that the PIR protocol is).

Private multiparty sampling can be used as a building block in a wide range of private approximation scenarios. For instance, it can be used in a straightforward way to obtain communication-efficient approximations for multiparty set intersection. Generalizing a two-party protocol of [13], if we let  $f$  be the bitwise AND function, the intersection size can be efficiently approximated (up to a small additive error) by making multiple invocations of the sampling primitive and outputting the fraction of 1's in the outputs. Private sampling can also be used, following the two-party techniques of [20], to obtain polylog-communication private approximation of the  $\ell^2$ -norm of the sum of the  $M$  inputs.

### 3.1 Oblivious Transfer with Distributed Receiver

Towards implementing private multiparty sampling, we introduce and study a distributed variant of oblivious transfer which is of independent interest. (This primitive can be used as a basic building block for sublinear-communication multiparty protocols, generalizing the protocol compiler from [27] to the multiparty case. Details are omitted due to space.)

<sup>2</sup> The protocol from [12] is described for the special case where  $f$  is the exclusive-or function but can be generalized (as was done in [13,20]) to arbitrary functions  $f$ . Our discussion is quite insensitive to the particular choice of  $f$  and in fact applies also to the simpler “distributed OT” primitive defined in Section 3.1.



In distributed OT, the role of the receiver is distributed between  $M$  parties. (A very different variant of distributed OT was considered in [29].) Specifically, a large database  $x$  of  $N$  entries is held by a distinguished party, say  $P_1$ , who functions as a sender. The entries of the database are indexed by some finite group of size  $N$ , which is taken to be  $\mathbb{Z}_N$  by default. The index  $n$  of the entry to be retrieved is distributed between the  $M$  parties in an additive way. That is,  $n = \sum_{m=1}^M n_m$ , where each  $n_m$  is a local input of  $P_m$  and addition is taken over the underlying group. At the end of the protocol, some distinguished party, say  $P_M$ , should learn the selected entry  $x_n$ . More formally, we define *distributed-receiver oblivious transfer* (or distributed OT for short) as an  $(M - 1)$ -private protocol for the following  $M$ -party functionality.

**Definition 1 (Functionality  $\text{DistOT}_G$ ).** *Let  $G$  be a finite group of size  $N$ . The functionality  $\text{DistOT}_G$  is defined as follows:*

- *Inputs: Each party  $m$ ,  $1 \leq m \leq M$ , holds a group element  $n_m \in G$ . The first party  $P_1$  additionally holds a database  $x = (x_n)_{n \in G}$ .*
- *The last party  $P_M$  outputs  $x_{n_1+\dots+n_M}$ . Other parties have no output.*

### 3.2 Private Multiparty Sampling from Distributed OT

We now present efficient black-box construction of private multiparty sampling protocols from distributed OT. We start by formally defining the sampling functionality induced by  $f$ .

**Definition 2 (Functionality  $\text{Sample-f}$ ).** *Let  $f$  be an  $M$ -party functionality. (The functionality  $f$  is a deterministic or randomized mapping from  $M$  inputs to  $M$  outputs.) The randomized functionality  $\text{Sample-f}$  is defined as follows:*

- *Inputs: Each party  $m$ ,  $1 \leq m \leq M$ , holds a database  $x^m = (x_n^m)_{n \in [N]}$ .*
- *The functionality picks a secret, uniformly random index  $r \in [N]$  and outputs  $f(x_r^1, x_r^2, \dots, x_r^M)$ .*

We start by handling the easier case where  $f$  is the identity function, outputting the concatenation of its  $M$  inputs. We denote the resulting sampling functionality by  $\text{Sample-ID}$ . In this case, we can use the following reduction to  $\text{DistOT}_G$  where  $G$  is an arbitrary group of size  $N$ . In the following, we arbitrarily identify elements of  $G$  with indices in  $[N]$ .

#### Reducing $\text{Sample-ID}$ to $\text{DistOT}$

1. Each party  $P_m$  picks a random group element  $r_m \in_R G$ .
2. In parallel, the parties make  $M$  calls to  $\text{DistOT}$ , where in call  $i$  party  $P_i$  acts as sender with database  $x^i$  and every party  $P_m$  (including  $P_i$ ) lets  $n_m = r_m$ . As a result, party  $P_M$  obtains the values  $x_{r_1+\dots+r_M}^m$  for  $1 \leq m \leq M$  and sends them to all parties.
3. Each party outputs the  $M$  values received from  $P_M$ .

The correctness and privacy of the above reduction are straightforward to verify.

We now turn to the question of obtaining **Sample-f** from **DistOT**, for a general functionality  $f$ . We start by observing that **Sample-f** can be efficiently reduced to a simpler (randomized) functionality **Sample-AS**, where **AS** (for “additive sharing”) outputs an  $M$ -tuple of strings that are random subject to the restriction that their exclusive-or is the concatenation of the  $M$  inputs.

**Proposition 1.** *For any polynomial-time computable  $M$ -argument function  $f$  there is a constant-round black-box  $(M - 1)$ -private reduction of **Sample-f** to **Sample-AS** and 1-out-of-2 OT.*

**Proof (sketch):** The reduction proceeds by first invoking **Sample-AS** to obtain an additively shared representation of the inputs to  $f$ , and then running a general-purpose constant-round protocol (based on OT) to compute  $f$  from these shares. For the latter one can use the protocol of Beaver et al. [2]. ■

In the above reduction, the OT primitive could be dispensed with, as it can be implemented from **Sample-AS** (using [9]).

Given the above, it suffices to reduce **Sample-AS** to **Sample-ID**. For simplicity, we restrict the attention to the case where each entry of a database  $x^m$  is a single bit. (The general case of  $\ell$ -bit entries can be handled analogously.) A natural approach that comes to mind is to let each party  $P_m$  mask every bit of  $x^m$  with a random bit  $b_m$ , invoke **Sample-ID** on the resulting masked databases, and then use a private computation of a (randomized) linear function to convert the masked entries  $x_r^m \oplus b_m$  together with the masks  $b_m$  into the required additive sharing. This approach fails for the following reason: an adversary corrupting  $P_m$  can learn both  $x_r^m \oplus b_m$  (from the output of **Sample-ID**) and the mask  $b_m$ , which together reveal  $x_r^m$  and thus (together with  $x^m$ ) give partial information about  $r$ . This is not allowed by the ideal functionality **Sample-AS**. Other variants of this approach fail for similar reasons.

To avoid the above problem, we must generate the masks in a completely distributed way. We achieve this by using **DistOT** over the group  $G' = G \times Z_2$ :

**Reducing Sample-AS to DistOT**

1. Each party  $P_m$  prepares an extended database  $(x')^m$  of size  $2N$  such that for each  $n' = (n, b) \in G'$  we have  $(x')_{n'}^m = x_n^m \oplus b$ . In addition, each  $P_m$  picks a random group element  $r_m \in_R G$  and  $M$  random bits  $b_{m,m'}, 1 \leq m' \leq M$ .
2. In parallel, the parties make  $M$  calls to **DistOT** $_{G'}$ . In call  $i$  party  $P_i$  acts as sender with database  $(x')^i$  and every party  $P_m$  (including  $P_i$ ) lets  $n'_m = (r_m, b_{m,i})$ . As a result, party  $P_M$  obtains the values  $(x')_{(r_1, b_{1,m}) + \dots + (r_M, b_{M,m})}^m = x_{r_1 + \dots + r_M}^m \oplus (b_{1,m} \oplus b_{2,m} \oplus \dots \oplus b_{M,m})$  for  $1 \leq m \leq M$ .
3. Each party  $P_m, m < M$ , outputs the  $M$ -tuple  $(b_{m,1}, b_{m,2}, \dots, b_{m,M})$ . For  $m = M$ , party  $P_M$  outputs the exclusive-or of this  $M$ -tuple with the  $M$ -tuple obtained in Step 2 above.

**Proposition 2.** *The reduction described above is an  $(M - 1)$ -private black-box reduction from **Sample-AS** to **DistOT**. Moreover, the reduction is totally non-interactive.*

Combining Propositions 1 and 2 yields an efficient black-box reduction from Sample-f to DistOT.

### 3.3 Implementing Distributed OT

It remains to implement DistOT. We mainly focus on general constructions based on 1-out-of- $n$  OT (equivalently, PIR [28,9]). For  $n \in G$ , we use  $x \ll_G n$  to denote the database  $x'$  obtained from  $x$  by applying the permutation induced by adding  $n$  to each index. That is,  $x'_{n'} = x_{n'+n}$ , where addition is in the group  $G$ . Note that in the default case where  $G = \mathbb{Z}_N$ , the notation “ $\ll$ ” corresponds to the usual notation of a cyclic shift to the left. When there is no ambiguity or when the choice of the group does not matter, we omit the group subscript.

#### A black-box construction of DistOT using one-round OT

A one-round OT can be specified by a randomized query algorithm  $Q(n, \rho)$  (where  $\rho$  is the receiver’s secret randomness), an answering algorithm  $A(x, q)$  and a reconstruction algorithm  $R(a, \rho)$ . (The security parameter  $k$  is implicit in this notation.) The reduction proceeds as follows. Each party  $P_m$  sends an OT query pointing to its input  $n_m$  to the sender  $P_1$ . Each such query can be used to “obliviously shift” the database  $x$  by the amount  $n_m$ ; more precisely, the  $n$ -th entry of a shifted database  $y$  is simply the answer to the OT query on  $y \ll n$ . The result of each such oblivious shift may be viewed as being encrypted using the key owned by the originator of the OT query. At the end of the  $M - 1$  oblivious shifts, the sender holds an  $(M - 1)$ -iterated encryption of  $x \ll (\sum_{m=1}^M n_m) - n_1$ . The  $(n_1)$ -th entry  $x_{n_1+\dots+n_M}$  can be recovered by passing its iterated encryption between the parties, letting each peel off its own layer of encryption using the OT reconstruction function. (See Figure 1)

1. Each party  $P_m, m > 1$ , picks an OT query  $q_m = Q(n_m, \rho_m)$ , and sends it to  $P_1$ .
2.  $P_1$  initializes  $a^{M+1} := x$ .
3. For  $i = M$  downto 2, party  $P_1$  lets  $a^i$  be a database of  $N$  entries defined by  $a^i_n = A(a^{i+1} \ll_G n_i, q_i)$ . It then sends  $b_1 = a^2_{n_1}$  to  $P_2$ .
4. For  $i = 2$  to  $M - 1$ , party  $P_i$  lets  $b_i = R(b^{i-1}, \rho_i)$  and sends  $b_i$  to  $P_{i+1}$ .
5. Party  $P_M$  outputs  $b_M = R(b^{M-1}, \rho_M)$ .

Fig. 1. A black-box reduction of DistOT to one-round OT

**Proposition 3.** *The reduction described in Figure 1 is  $(M - 1)$ -private.*

**Proof (sketch):** Correctness is easy to verify. For privacy, we briefly sketch a formal construction of a simulator. The simulator is given the inputs of corrupted parties and, possibly, the output  $x_{n_1+\dots+n_M}$ . It simulates the message sequence  $b_i$  by iteratively applying the OT simulator starting from either the actual output

(if  $P_M$  is corrupted) or a default output otherwise. The output of each iteration is used for the next iteration, along with either an actual input  $n_i$  (if  $P_i$  is corrupted) or a default input (if  $P_i$  is uncorrupted). This simulation process produces all messages  $b_i$  along with the local inputs  $\rho_i, q_i$  in reverse order. ■

The complexity of the above reduction depends on the number of parties  $M$  and the relation between the size  $\ell'$  of the answers of the OT protocol to the length of the database entries  $\ell$ . Ideally, we have  $\ell' = \ell + k \cdot \text{polylog}(N)$ , where  $k$  is the security parameter. (Such an OT protocol can be based on the Damgård-Jurik encryption scheme [10,25].) In this case, the complexity of the resulting DistOT protocol (on top of the length of the OT queries) is  $\ell + Mk \cdot \text{polylog}(N)$ . Thus, the protocol can be applied also for non-constant  $M$ . When the number of parties  $M$  is viewed as constant, the DistOT protocol can be made communication-efficient even if, say,  $\ell' = \text{poly}(\ell, k) \cdot N^{0.9}$ . (By “communication-efficient,” we mean that the dependence on  $N$  can be reduced to  $\text{poly}(\ell, k) \cdot N^\epsilon$  for an arbitrary  $\epsilon > 0$ .) Such an OT protocol can be based on an arbitrary homomorphic encryption scheme [23,31].

## 4 Private Approximation of Vector Combinations

In this section, towards further reducing the communication complexity, we consider as approximation functions any of a wide array of natural “summary” functions of the combined data vector  $y$  (e.g., the  $\ell^2$ -norm or an approximate  $t$ -term Fourier representation). Specifically, we consider  $M \geq 3$  parties holding length- $N$  vectors  $x^1, \dots, x^m$  who, ideally, want to compute (and are willing to disclose to the other parties) the componentwise sum  $y = \sum_m x^m$  of their vectors or the componentwise minimum  $y = \bigwedge_m x^m$  of their vectors. The actual output computed is an approximate size- $t$  summary  $Y$  for  $y$  (e.g., the  $\ell^2$ -norm, or an approximate  $t$ -term Fourier representation).

We exploit the fact that the entire (long) vector  $y$  may be leaked in order to obtain simple private protocols for the above problems, in which the communication and the number of public-key operations are at most polynomial in the size of the small approximation and polylogarithmic in  $N$ . In contrast, most previous protocols for sublinear-communication secure approximation (including the results of Section 3) require roughly as many public-key operations as the size of the entire database (given the current state of the art of PIR).

### 4.1 Vector Sums

**Proposition 4.** *Suppose parties  $P_1, P_2, \dots, P_M$  hold length- $N$  vectors  $x^1, x^2, \dots, x^M$ . Let  $y = \sum_m x^m$  be the componentwise sum. There is a protocol for generating a Gaussian random variable with mean zero and variance  $\sum_n y_n^2$  that leaks (to any subset of the parties) no more than  $y$ , requires local computation  $NM^{O(1)}$ , communication  $M^{O(1)}$ , and  $O(1)$  rounds of interaction.*

**Proof:** If each component  $r_n$  of a vector  $r$  is a unit normal random variable, then  $\sum_n r_n y_n$  is a Gaussian random variable  $Y$  with mean zero and variance equal to our desired value,  $\sum_n y_n^2$ . In particular,  $Y$  together with  $r$  leak nothing else about the inputs  $x^m$  beyond what is implied by their sum  $y$ . The sum  $\sum_n r_n y_n$  can in turn be computed by first letting each party compute a local sum  $s^m = \sum_n r_n x_n^m$  and then using a standard  $(M - 1)$ -private protocol for adding up the  $M$  (short) integers  $s^m$ . The protocol is described in Figure 2.

The communication of the protocol in Figure 2 is as claimed, because the secure-sum protocol is applied to  $M$  numbers  $s^m$ , and not length- $N$  vectors. ■

**Sketch Sum**

1. The parties agree on pseudorandom Gaussian random vector  $r$  in the clear.
2. Party  $m$  receives vector  $x^m$  as input.
3. The parties individually compute sketches  $s^m = \sum_n r_n x_n^m$ .
4. The parties use a secure-sum sub-protocol to compute  $\sum_m s^m = \sum_m \sum_n r_n x_n^m = \sum_n r_n \sum_m x_n^m = \sum_n r_n y_n$ , where  $y = \sum_m x^m$  is the componentwise sum of the parties' input vectors.

**Fig. 2.** A protocol for computing an additive sketch

**Sketch Min**

1. The parties agree on pseudorandom exponential random vector  $r$  in the clear.
2. Party  $m$  receives vector  $x^m$  as input.
3. The parties individually compute sketches  $s^m = \bigwedge_n r_n x_n^m$ .
4. The parties use a secure sub-protocol to compute  $\bigwedge_m s^m = \bigwedge_m \bigwedge_n r_n x_n^m = \bigwedge_n r_n \bigwedge_m x_n^m = \bigwedge_n r_n y_n$ , where  $y = \bigwedge_m x^m$  is the componentwise minimum of the parties' input vectors.

**Fig. 3.** A protocol for computing a minimum sketch

## 4.2 Vector Minima

We generalize the above protocol to the componentwise minimum instead of sum. Also, instead of approximating the quantity  $\sum_n y_n^2$ , we approximate the harmonic mean, or its inverse,  $\sum_n y_n^{-1}$ . See, e.g., [7] for example uses in algorithms of estimating the parameter of an exponential random variable.

**Proposition 5.** *Suppose parties  $P_1, P_2, \dots, P_M$  hold length- $N$  positive-valued vectors  $x^1, x^2, \dots, x^M$ . Let  $y = \bigwedge_m x^m$  be the componentwise minimum. There is a protocol for generating an exponential random variable with parameter  $(\sum_n y_n^{-1})$  that leaks (to any subset of the parties) no more than  $y$ , requires local computation  $NM^{O(1)}$ , communication  $M^{O(1)}$ , and  $O(1)$  rounds of interaction.*

**Proof:** It is known that, if each component  $r_n$  of a vector  $r$  is a unit exponential random variable, then  $\sum_n r_n y_n$  is an exponential random variable  $Y$  with parameter equal to our desired value of  $\sum_n y_n^{-1}$ . In particular,  $Y$  together with  $r$  leak no more than  $y$ . The parties use the protocol of Figure 3. The subproblem for which a secure protocol is needed is computing the minimum of  $M$  short integers, for which efficient  $(M - 1)$ -private protocols exist (e.g., using the general-purpose constant-round protocol of [2]). ■

## Acknowledgments

Yuval Ishai was supported by grants 36/03 and 1310/06 from the Israel Science Foundation; Tal Malkin by NSF grant CCF-0347839; Martin Strauss by NSF grants DMS-0510203 and DMS-0354600; and Rebecca Wright by NSF grant CCR-0331584. We thank Stillian Stoev for helpful discussions.

## References

1. Alon, N., Gibbons, P., Matias, Y., Szegedy, M.: Tracking join and self-join sizes in limited storage. In: Proc. Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 10–20. ACM Press, New York (1999)
2. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols. In: Proc. 22th ACM STOC, pp. 503–513 (1990)
3. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 404–414. Springer, Heidelberg (1999)
4. Candès, E., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory* 52(2), 489–509 (2006)
5. Canetti, R.: Security and composition of multiparty cryptographic protocols. *J. Cryptology* 13(1), 143–202 (2000)
6. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: Proc. 36th IEEE FOCS, pp. 41–50. IEEE Computer Society Press, Los Alamitos (1995)
7. Cohen, E.: Size-estimation framework with applications to transitive closure and reachability. *J. Computer and System Sciences* 55(3), 441–453 (1997)
8. Cormode, G., Muthukrishnan, S.: Estimating dominance norms of multiple data streams. In: Proc. 11th European Symposium on Algorithms, pp. 148–160 (2003)
9. Di Crescenzo, G., Malkin, T., Ostrovsky, R.: Single database private information retrieval implies oblivious transfer. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 122–138. Springer, Heidelberg (2000)
10. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier’s probabilistic public-key system. *Public Key Cryptography* 119–136 (2001)
11. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. *Communications of the ACM* 28, 637–647 (1985)
12. Feigenbaum, J., Ishai, Y., Malkin, T., Nissim, K., Strauss, M., Wright, R.: Secure multiparty computation of approximations. *ACM Transactions on Algorithms* 2(3), 435–472 (2005). An earlier version of this paper appeared in ICALP 2001 (2001)

13. Freedman, M., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
14. Gentry, C., Ramzan, Z.: Single-database private information retrieval with constant communication rate. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 803–815. Springer, Heidelberg (2005)
15. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. *J. Computer and System Sciences* 60(3), 592–692 (1998) A preliminary version appeared in 30th STOC (1998)
16. Gilbert, A., Guha, S., Indyk, P., Kotidis, Y., Muthukrishnan, S., Strauss, M.: Fast, small-space algorithms for approximate histogram maintenance. In: Proc. 34th ACM STOC, pp. 389–398. ACM Press, New York (2002)
17. Goldreich, O.: Secure multi-party computation (working draft, version 1.1) (1998) Available at <http://philby.ucsd.edu/CRYPTOLIB/BOOKS/oded-sc.html>
18. Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: Proc. 19th ACM STOC, pp. 218–229. ACM Press, New York (1987)
19. Goldreich, O., Vainish, R.: How to solve any protocol problem—an efficiency improvement. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 73–86. Springer, Heidelberg (1988)
20. Indyk, P., Woodruff, D.: Private polylogarithmic approximations and efficient matching. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 245–264. Springer, Heidelberg (2006)
21. Killian, J.: Founding cryptography on oblivious transfer. In: Proc. 20th ACM STOC, pp. 20–31 (1988)
22. Kushilevitz, E., Mansour, Y.: Learning decision trees using the fourier spectrum. In: Proc. 23th ACM STOC, pp. 455–464. ACM Press, New York (1991)
23. Kushilevitz, E., Ostrovsky, R.: Replication is NOT needed: SINGLE database, computationally-private information retrieval. In: Proc. 38th IEEE FOCS, pp. 364–373. IEEE Computer Society Press, Los Alamitos (1997)
24. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 177–206. Springer, Heidelberg (2000)
25. Lipmaa, H.: An oblivious transfer protocol with log-squared communication. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)
26. Mann, E.: Private access to distributed information. Master’s thesis, Technion - Israel Institute of Technology, Haifa (1998)
27. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: Proc. 33th ACM STOC, pp. 590–599. ACM Press, New York (2001)
28. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: Proc. 31st ACM STOC, pp. 245–254. ACM Press, New York (1999)
29. Naor, M., Pinkas, B.: Distributed oblivious transfer. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, Springer, Heidelberg (2000)
30. Rabin, M.O.: How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University (1981)
31. Stern, J.P.: A new and efficient all-or-nothing disclosure of secrets protocol. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 357–371. Springer, Heidelberg (1998)

# Constant-Round Private Database Queries

Nenad Dedic<sup>1,\*</sup> and Payman Mohassel<sup>2,\*\*</sup>

<sup>1</sup> Department of Computer Science, Boston University  
nenad@cs.bu.edu

<sup>2</sup> Department of Computer Science, University of California Davis  
mohassel@cs.ucdavis.edu

**Abstract.** We consider several private database query problems. The starting point of this work is the *element rank* problem: the server holds a database of  $n$  integers, and the user an integer  $q$ ; the user wishes to find out how many database records are smaller than  $q$ , without revealing  $q$ ; nothing else about the database should be disclosed. We show a non-interactive communication-efficient solution to this problem. We then use it to solve more complex private database queries: range queries, range queries in plane and higher-dimensional generalizations of element rank. We also show an improved solution to the  $k^{\text{th}}$  ranked element problem [1], and a solution to *private keyword search* [9] using weaker assumptions than those of [9]. All our solutions assume semi-honest adversarial behaviour.

## 1 Introduction

*Private information retrieval* (PIR) allows a user to query the server's database, without revealing information about the query. If the user cannot find out more information than required, then we have *symmetrically private information retrieval* (SPIR). Efficient solutions currently exist for limited types of queries: retrieval by address (the first considered PIR problem [7,14]), retrieval by keyword [9]. More complex queries can be processed as well, but less efficiently [6,17] or using relatively strong computational assumptions [13]. It is therefore desirable to find efficient solutions to more specific private database query problems.

For example, a new bidder in an auction might be allowed to learn how his bid ranks compared to other bids, but unwilling to disclose the bid to the auction server. Or consider a user who wants to compare his score against the database of an online testing service provider, while keeping his score private. These are examples of the *element rank problem*. As another motivating example, consider a driver who wishes to get a list of all fast-food joints in a city block, but does not want to disclose his location. This is an example of *2-d range retrieval*.

In this work we mainly consider the (generalization of) element rank problem. Based on this we show solutions to several other problems (e.g. range retrieval, 2-d range retrieval, shared element rank). We seek to minimize two important efficiency measures: *communication complexity* and *round complexity*. Minimal round complexity is noteworthy because it allows for easier integration with other protocols; we exemplify this

---

\* Part of this work was done while the author was visiting the IPAM institute at UCLA.

\*\* Part of this work was done while the author was visiting the IPAM institute at UCLA. This work was partially supported by the Packard Foundation.



by using the non-interactive nature of our basic rank protocol to solve other problems. Note that by *non-interactive*, we mean protocol with *one round* of communication between the parties.

## 1.1 Contributions

We solve the following problems, in the semi-honest model:

*Interval labeling problem.* As the main building block, we design a non-interactive low-communication solution to the following problem. Two players are involved: the user whose input is an integer  $q$ , and the server whose input is a sorted sequence of integers  $x_1, \dots, x_n$ , defining  $n + 1$  intervals, labeled with strings  $v_0, \dots, v_n$ . The user wishes to find out the label of the interval to which  $q$  belongs. Our most efficient solution can be based on homomorphic encryption and PIR [14].

*Element rank; generalizations.* We then show how to use interval labeling to efficiently solve the rank problem, and generalizations of interval labeling (for example, *rectangle labeling* where the server subdivides the plane in rectangles and labels each rectangle).

*Equivalence of PIR and PKS.* On a more theoretical note we show that, somewhat surprisingly, non-interactive PIR alone suffices to implement non-interactive PKS. Communication complexity is (in asymptotic terms) almost the same as that of the underlying PIR. Previous solutions known to us use more specific algebraic assumptions (e.g. [9] requires homomorphic encryption). However, in a practical scenario, the solution of [9] would likely be preferable because of greater efficiency. We note that as a corollary we obtain that the interval labeling problem and its generalizations can be efficiently implemented using PIR alone.

*Secure computation of  $k^{\text{th}}$ -ranked element.* In [1], Aggrawal et al. design a protocol for secure computation of the  $k^{\text{th}}$ -ranked element (e.g. the median) in the union of private databases held by two parties. We use our new non-interactive protocol for secure computation of rank of an element in a database as a subprotocol, and design more efficient protocols for this task. More specifically, we design a new protocol with  $\log \log(k)$  rounds of interaction between the two parties, as opposed to  $\log(k)$  rounds of interaction required by the protocol of [1].

*Constant-round range queries; on a line or in a plane.* The server holds a database of points on a line (or, for example, on a plane). The user holds a high point and a low point on that line (or an axis-aligned rectangle in the plane). As the output of the protocol the user should only learn the points in the database that belong to the range defined by his input values, i.e., all the points in the database that lie between the high and low points (or all the points that lie inside user's rectangle). We design constant-round protocols for this problem, with communication linear in the number of retrieved points. Our protocols provide privacy for both user and the server.

## 1.2 Related Work

Private information retrieval (PIR), introduced by Chor, et al. in [7] is the prototypical problem of private database query evaluation. It requires that the user with input  $i$  learn  $i$ -th entry of some database  $D$ . Security requirement is that  $i$  remain hidden to everyone

but the user. Another, essential, requirement is that the total communication be strictly less than the size of  $D$  (else the problem is trivially solved by sending  $D$ ). The solution given in [7] mandates statistical security, and requires multiple non-communicating servers holding copies of  $D$ . The multiple-server requirement was lifted in [14], at the necessary expense of downgrading to computational security. [10] introduce *symmetric* PIR (SPIR), where server privacy is required: the user must learn nothing beyond  $D[i]$ . SPIR can be seen as communication-efficient oblivious transfer [20]. The works of [18,2] show how to convert any PIR to SPIR, at little cost. Many solutions to the PIR problem have appeared since [4,5,16], improving the efficiency, or weakening/changing the underlying hardness assumptions. Most of these solutions share an attractive feature – they are *non-interactive* in the sense that they use only a single communication round.

General multi-party computation (e.g. see [12] and [11]) can be used to solve private database query problems, but not very efficiently. It is therefore of interest to seek more efficient solutions, even if they are more specialized. We focus specifically on minimizing communication and interaction. *Communication-efficient secure function evaluation* [17] gives a protocol for secure computation of circuits with lookup tables. Such a protocol takes any oblivious Turing machine with RAM access and turns it into a private two-party protocol. While this can be used to implement a communication-efficient element rank protocol, the construction is not concerned with optimizing number of rounds of *interaction* between the parties. *Computing branching programs on encrypted data* is introduced in an independent work [13]. This work considers queries which can be represented as branching programs and gives a non-interactive solution to this problem. The construction given in [13] is more general than our private element rank protocol and can be used to implement such a task. But, we use different techniques, and by doing so can depend on more *general* and *weaker* assumptions. Furthermore, some of the protocols we study in this paper such as private range queries (see section 4) cannot be captured efficiently using branching programs.

*Private keyword search* (PKS) [9,6] is a natural extension of SPIR where the server holds a set of key-value pairs  $(k_i, v_i)$ , and the user wishes to find a value associated to some keyword  $k$  if it exists. [6] in fact show how to privately query any server-held data structure, using multiple interaction rounds. *Private keyword search on streaming data* [19] considers a different model, where tagged documents come in streaming, and an untrusted computer keeps, in an encrypted form, only those which have certain user-specified combination of tags. This combination remains hidden to the untrusted computer. *Oblivious binary search and secure range queries* [8] deals with some extensions to PIR (e.g. retrieving entries at addresses  $i, i + 1, \dots, j$ ), as well as with computing element rank, but using a logarithmic number of rounds; this is then applied in a range retrieval protocol, where the user whose input is  $(a, b)$  finds out all database entries with *values* between  $a$  and  $b$ .

A closely related problem of non-interactive secure computation is considered in [3]. Their setting involves a public circuit  $C$ , Alice with private input  $x$  and Bob with private input  $y$ . After a single round of interaction Alice should learn  $C(x, y)$  and Bob should learn nothing. They solve this problem for any polynomial-size circuit  $C$ . Communication complexity, however, depends polynomially on the sizes of  $x, y$ , even if the size of  $C(x, y)$  is very small.

## 2 Preliminaries

*Notation and conventions.* We interpret binary strings as integers when convenient, with the most significant bit to the left.  $[N]$  denotes the set  $\{0,1,2,\dots,N\}$ , and  $[N)$  denotes  $[N - 1]$ .  $|x|$  denotes the length of  $x$ .  $\epsilon$  denotes the empty string. We write  $x||y$  to denote the concatenation of strings  $x$  and  $y$ . We write  $x^-$  to denote  $x$  with the last symbol dropped (e.g.  $aaba^- = aab$ ), and for a binary string  $x$  we write  $x'$  to denote  $x$  with the last bit flipped (e.g.  $1100' = 1101$ ). We write  $p \preceq s$  to denote that  $p$  is a prefix of  $s$ ; and  $p \prec s$  to denote  $p \preceq s \wedge p \neq s$ .

### 2.1 Oblivious and Succinct Computation

In this section we introduce *oblivious computation*, a useful abstraction for certain non-interactive cryptographic tasks. This notion appears implicitly in [21] and more explicitly in [3]. We systematize and make explicit its important features. Intuitively, oblivious computation involves two parties: the *user Alice* holding her input  $x$ , and the *server Bob* holding his input  $y$ . Some function  $f$  is given publicly, and Alice wishes to learn  $f(x, y)$ . Bob should learn nothing, and Alice should learn nothing beyond  $f(x, y)$ . Alice will send some query  $ex = Q(x)$  (which hides  $x$ ) to Bob, who will produce an answer  $a = A(ex, y)$  and send it back to Alice. Alice will then decode this answer to get  $D(a) = f(x, y)$ . Formally:

**Definition 1.** We say that a quadruple of PPT algorithms  $(G, Q, A, D)$  obliviously computes a function  $f(x, y)$  if for  $(PK, SK)$  generated by  $G(1^k)$ :

**Correctness holds:**  $D(SK, A(PK, Q(SK, x), y)) = f(x, y)$ .

**Query secrecy holds:** for any  $x, x'$ , the distributions  $Q(SK, x)$  and  $Q(SK, x')$  are computationally indistinguishable

**Answer security holds:** some PPT algorithm  $S$  simulates  $A(PK, Q(SK, x), y)$  given only  $(PK, x, f(x, y))$ .

$G$  is called the instance generator,  $Q$  the query computer,  $A$  the answer computer and  $D$  the answer decoder.

This definition naturally corresponds to one-round secure two-party computation, where only Alice gets the output. Any polynomial-sized function  $f$  can be obliviously computed using one-round oblivious transfer and Yao’s garbled circuits [3]. But we will in fact consider a more stringent setting, where no general solution is known. The server holds a large database  $X = (x_1, \dots, x_n)$  and the user holds a short query  $q$ . They are interested in obliviously computing some function  $f(q, X)$  whose output size is independent of  $n$ , while minimizing communication. Formally:

**Definition 2.** We say that a quadruple of PPT algorithms  $(G, Q, A, D)$  succinctly computes  $f(q, X = (x_1, \dots, x_n))$  if:

**Obliviousness holds:**  $(G, Q, A, D)$  obliviously computes  $f$

**Succinctness holds:** there is a polynomial  $p$  such that

$$|Q(SK, q)| + |A(PK, Q(SK, q), X)| \leq p(\log n) \text{ for sufficiently large } n.$$

Succinct computation is a natural umbrella definition for non-interactive cryptographic tasks with a communication restriction. Some examples, which we use in this work, are:

**Private information retrieval** is succinct computation of the function

$$f(i, (x_1, \dots, x_n)) = x_i.$$

Solutions are given, for example, in [5][16]. Note that most definitions of PIR impose no limit on the number of rounds. But almost all PIR protocols from the literature are in fact non-interactive (recall, this means that they require a single round), and therefore conform to the definition of succinct computation. This can be used to good effect in protocols where PIR is a building block, so for purposes of this work we take PIR to be non-interactive.

**Private keyword search** [9] is succinct computation of the function

$$PKS(w, (x_1, \dots, x_n, v_1, \dots, v_n)) = \begin{cases} v_a, & \exists a \ w = x_a \\ \perp, & \text{otherwise.} \end{cases}$$

We also use *contiguous PIR* – a natural generalization of PIR, where a contiguous block of  $s$  records is retrieved, while communicating  $\Theta(s)$  bits. Note that the server must necessarily learn  $s$ , simply by counting the number of bits sent. The corresponding functionality is

$$ContPIR((i, s); (x_1, \dots, x_n)) = ((x_i, \dots, x_{i+s-1}); s).$$

[8] give a solution to this problem.

### 3 Computing Element Rank and Related Functions

We consider the following problem, which is at the heart of all the protocols in this paper.

*Problem 1 (Interval Labeling).* Two parties, the user Alice and the server Bob are involved. Let  $L = 2^l - 1$ ,  $n$  and  $m$  be public integer parameters. **Parties’ inputs** are as follows:

**Server** holds  $x_1, \dots, x_n \in [L]$  and  $v_0, \dots, v_n \in \{0, 1\}^m$ .  $x_1, \dots, x_n$  is an increasing sequence of numbers subdividing the interval  $[L]$  into  $n + 1$  disjoint sub-intervals  $(x_0, x_1], (x_1, x_2], \dots, (x_n, x_{n+1}]$  (where we take by convention  $x_0 = -\infty$ ,  $x_{n+1} = +\infty$ ). In each interval  $(x_i, x_{i+1}]$  the server writes a string label  $v_i$ .

**User** holds an integer  $q \in [L]$ .

**The output** which the user wishes to find out is the label  $v_i$  of the interval  $(x_i, x_{i+1}]$  to which  $q$  belongs. **Security** requirements are: 1. Server learns nothing, 2. User learns nothing except  $v_i$ . **Efficiency** requirements are: 1. computations must be polynomial time, 2. communication complexity must be  $\text{polylog}(nl)$ .

#### 3.1 Succinctly Computing Interval Labeling and Element Rank

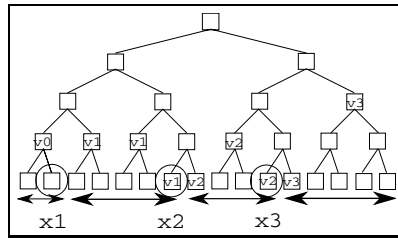
Let us see how to reduce succinct computation of interval labeling to private keyword search (PKS). It will be shown then that this implies a solution to element rank. An obvious way, but too expensive one, would be to create a PKS database of  $L$  entries:

for every  $y \in [L]$  insert  $(y, v_i)$  into the database where  $i$  is such that  $y \in (x_{i-1}, x_i]$ . Searching for the keyword  $q$  in this database would clearly return the correct answer  $v_i$ . We now show how the database size can be reduced from  $L$  to  $2ln$ , by inserting only certain prefixes of  $x_i$ 's.

We refer the reader to Section 2 to review the notation and conventions. In this section we use additional notation: for a binary string  $s \in \{0, 1\}^j$ , let  $\bar{s}$  denote the interval of integers  $[s|0^{l-j}, s|1^{l-j}]$ . Also, for user's query  $q$ , let  $q_i$  denote its  $i$ -th bit.

*Database construction.* The main observation of this section is the following. Consider  $w \in \{0, 1\}^m$  and  $i$  such that  $\bar{w} \subseteq (x_{i-1}, x_i]$  and  $\bar{w}' \not\subseteq (x_{i-1}, x_i]$ . Then  $(w, v_i)$  can be added to the PKS database without introducing any ambiguity. Indeed, all integers beginning with the prefix  $w$ , fall into the same interval  $(x_{i-1}, x_i]$ , and they therefore share the label  $v_i$ . Moreover, for any predecessor  $u \prec w$ , all integers beginning with  $u$  span more than one interval.

This is illustrated in the following picture. Leaves of the tree represent the integers of  $[L]$ , with 0 in the left, and increasing towards right. Database values  $x_i$  are circled, and intervals  $(x_{i-1}, x_i]$  are indicated by arrows. Nodes corresponding to  $w$ 's described above are labeled with their corresponding  $v_i$ 's. In this example,  $L = 16$ ,  $n = 3$  and  $(x_1, x_2, x_3) = (1, 6, 10)$ .



Suppose now that  $D$  is formed by inserting all such  $(w, v_i)$  into it, i.e.

$$D = \{(w, v_i) \mid \bar{w} \subseteq (x_{i-1}, x_i] \wedge \bar{w}' \not\subseteq (x_{i-1}, x_i]\}.$$

Then  $D$  has the following useful properties (for proof please refer to the full version):

**Lemma 1.** *Let  $q \in (x_{i-1}, x_i]$ . Then there is exactly one prefix  $w \prec q$  for which  $(w, v_i) \in D$ . Every other prefix  $p \prec q$  ( $p \neq w$ ) is not contained as a keyword in  $D$ .*

*User's query.* On input  $q \in [L]$  the user computes private keyword search queries  $kwq_1, \dots, kwq_l$ , where  $kwq_j$  corresponds to the prefix  $p_j := q_1||q_2|| \dots ||q_j$ . Lemma 1 ensures that exactly one of these queries will return the correct  $v_i$ .

**Protocol Description.** The function corresponding to the interval labeling problem is

$$IL(q, (x_1, \dots, x_n, v_0, \dots, v_n)) = v_a, \text{ where } q \in (x_a, x_{a+1}].$$

Note that element rank

$$Rank(y, (x_1, \dots, x_n)) = |\{x_i \mid x_i < y\}|$$

is a special case of  $IL$  – use  $v_i = i$  for  $0 \leq i \leq n$ . We show how to obliviously compute  $IL$ , using private keyword search as the main building block. Let  $(PKSG, PKSQ, PKSA, PKSD)$  be the algorithms implementing succinct computation of  $PKS$ . Then we have the following

*Claim.* Scheme **I** (see below) succinctly computes  $IL$  (i.e. it solves Problem **I**).

*Proof. Sketch.* The main ingredient is the following simple claim.

*Claim.* Let  $\overline{w} \subseteq (x_{i-1}, x_i]$  and  $\overline{w'} \not\subseteq (x_{i-1}, x_i]$ . Then  $w \prec x_i \vee w \prec x_{i-1} \vee w' \prec x_i \vee w' \prec x_{i-1}$ .

*Proof.* Straightforward case analysis.  $\square$

Let  $D$  be the database obtained by the algorithm  $ILA$ . Using the above claim, it is easy to prove that  $(w, v_i) \in D \iff (\exists i) \overline{w} \subseteq (x_{i-1}, x_i] \wedge \overline{w'} \not\subseteq (x_{i-1}, x_i]$ . To see this, note that  $ILA$  iterates through the set  $S = \{w \mid (\exists i) w \prec x_i \vee w' \prec x_i\}$ ; but the above claim guarantees that it is sufficient to examine exactly the set  $S$ . This means that Lemma **I** holds for  $D$ . We are now ready to prove:

*Correctness.* By Lemma **I**, exactly one of the queries  $p_j$  has a matching entry  $(p_j, v_i) \in D$ , and  $i$  is such that  $q \in (x_{i-1}, x_i]$ .

*Privacy.* As said above, there is exactly one  $p_j$  with a matching database entry. All other  $p_i \prec q$  will return  $\perp$  when used in private keyword search on  $D$ . Since the order of the answers is shuffled, using privacy of PKS and standard hybrid arguments, it is straightforward to show the privacy of interval labeling. For lack of space we do not present the proof here.

*Efficiency.* Note that  $D$  contains at most  $2ln$  entries. Server computation is therefore  $O(ln)$ . Total communication, as well as user's computation is polylogarithmic in  $n$  (and linear in  $l$ ).

#### Scheme 1 (Interval Labeling)

$ILG(1^k)$ :

1.  $output(SK, PK) = PKSG(1^k)$

$ILQ(SK, q)$ :

1.  $kw_{q_1} \leftarrow PKSQ(SK, q_1)$ ,  
 $\dots$ ,  
 $kw_{q_l} \leftarrow PKSQ(SK, q_l \parallel \dots \parallel q_l)$
2.  $output(kw_{q_1}, \dots, kw_{q_l})$

$ILA(PK, (kw_{q_1}, \dots, kw_{q_l}), A = (v_0, x_1, v_1, \dots, x_n, v_n))$ :

1. for  $j = 1$  to  $l$ :
  - (a) for  $i = 1$  to  $n$ :
    - i. let  $w$  be the  $j$ -bit prefix of  $x_i$
    - ii. if  $\overline{w} \subseteq (x_{i-1}, x_i]$  and  $\overline{w'} \not\subseteq (x_{i-1}, x_i]$  then add  $(w, v_a)$  to  $D$
    - iii. if  $\overline{w} \not\subseteq (x_{i-1}, x_i]$  and  $\overline{w'} \subseteq (x_{i-1}, x_i]$  then add  $(w', v_b)$  to  $D$
2. if fewer than  $2ln$  entries were added, then pad  $D$  to  $2ln$  entries
3. for  $j = 1$  to  $l$ , compute  $ra_j = PKSA(PK, kw_{q_j}, D)$
4. randomly permute  $(ra_1, \dots, ra_l)$  and output the result

$ILD(SK, (a_1, \dots, a_k))$ :

1. find the unique  $a_i$  such that  $PKSD(SK, a_i) = v \neq \perp$ ; output  $v$

Note that the keywords which the algorithm  $ILA$  inserts in  $D$  need not be of the same length. PKS solutions do not support variable lengths, but it is easy to circumvent this problem: for example, a prefix-free encoding can be padded to some fixed length.

### 3.2 Extending to Higher Dimensions

Interval labeling can be naturally extended to labeling of (hyper-)rectangles in  $[L]^d$ . Here a point is defined by its  $d$  coordinates  $(x_1, \dots, x_d) \in [L]^d$ . Let  $X_1 = (x_1[1] < \dots < x_1[n]), \dots, X_d = (x_d[1] < \dots < x_d[n])$  be  $d$  lists of coordinates (for simplicity assume they are all of equal length  $n$ ). These lists then define  $(n+1)^d$  rectangles, labeled  $R_{0, \dots, 0}, \dots, R_{n, \dots, n}$ . For  $i_1, \dots, i_d \in [n]$  we define the rectangle  $R_{i_1, \dots, i_d} := \{(x_1, \dots, x_d) \mid \forall j \ x_j \in (x_j[i_j], x_j[i_{j+1}])\}$ . In each rectangle  $R_{i_1, \dots, i_d}$ , a string label  $z_{i_1, \dots, i_d}$  is written.

This construction, which we will call *d-dimensional rectangle labeling*, is uniquely defined by

$(X_1, \dots, X_d, Z = (z_{0, \dots, 0}, \dots, z_{n, \dots, n}))$ . We are interested in succinct computation of

$$RL_d(p = (x_1, \dots, x_d), (X_1, \dots, X_d, Z)) = z_{i_1, \dots, i_d} \text{ where } p \in R_{i_1, \dots, i_d}.$$

This problem can be solved by recursively using the succinct computation of interval labeling. Rather than write the general solution for any  $d$ , we will exemplify it using  $d = 4$ . This conveys the main idea clearly, but cumbersome notations associated with arbitrary  $d$  are avoided.

We will denote the  $d = 4$  coordinates with  $(x, y, u, v)$ . Server's input is  $X = (x_1, \dots, x_n), Y = (y_1, \dots, y_n), U = (u_1, \dots, u_n), V = (v_1, \dots, v_n)$  ( $x_i, y_i, u_i, v_i \in [L]$ ) and a table of values  $Z = (z_{0,0,0,0}, \dots, z_{n,n,n,n})$  ( $z_{i,j,k,l} \in \{0, 1\}^m$ ). User's input is a point  $p = (x, y, u, v) \in [L]^4$ . A scheme for succinct computation of  $RL_4$  follows.

#### Scheme 2

$RLG_4(1^k)$ :

1. *output* (PK, SK) =  $ILG(1^k)$

$RLQ_4(\text{SK}, q)$ :

1. *output* ( $ILQ(\text{SK}, x), ILQ(\text{SK}, y), ILQ(\text{SK}, u), ILQ(\text{SK}, v)$ )

$RLA_4(\text{PK}, (ex, ey, eu, ev), (X, Y, Z))$ :

1. *for*  $(i, j, k) \in [n]^3$  *let*  $D_{i,j,k} \leftarrow ILA(\text{PK}, eqv, (v_1, z_{i,j,k,1}, \dots, v_d, z_{i,j,k,n}))$
2. *for*  $(i, j) \in [n]^2$  *let*  $E_{i,j} \leftarrow ILA(\text{PK}, equ, (u_1, D_{i,j,1}, \dots, u_c, D_{i,j,n}))$
3. *for*  $i \in [n]$  *let*  $F_i \leftarrow ILA(\text{PK}, eqy, (y_1, E_{i,1}, \dots, y_b, E_{i,n}))$
4. *let*  $ans \leftarrow ILA(\text{PK}, eqx, (x_1, F_1, \dots, x_n, F_n))$
5. *output*  $ans$

$RLD_4(\text{SK}, a)$ :

1. *User outputs*  $ILD(\text{SK}, ILD(\text{SK}, ILD(\text{SK}, ILD(\text{SK}, a))))$

*Claim.* Scheme 2 succinctly computes the function  $RL_4$ .

## 4 Range Retrieval in 2 Dimensions

In this section we are dealing with range retrieval in 2-dimensions. For two points  $a = (x_a, y_a), b = (x_b, y_b)$  define the *rectangle*  $\langle a, b \rangle$  as  $\{(x, y) \mid x \in [x_a, x_b], y \in [y_a, y_b]\}$ . The *server*  $S$  holds a database  $D$  of  $n$  points  $p_1, \dots, p_n$  where  $p_i \in [0, L] \times [0, L]$  ( $L$  is

a public parameter). The user  $U$  holds two points  $q_1 = (qx_1, qy_1), q_2 = (qx_2, qy_2) \in [0, L) \times [0, L)$ , and he wishes to retrieve all the points  $p_i \in D$  which are in the rectangle  $\langle q_1, q_2 \rangle$ . The communication of the protocol should be proportional to the number of retrieved points  $|D \cap \langle q_1, q_2 \rangle|$ , and the server should learn only that number.

The basic idea behind the protocol is as follows. Assume for simplicity that the server's database contains no two points with equal either  $x$  or  $y$  coordinates, i.e.  $x_i \neq x_j$  and  $y_i \neq y_j$  for all  $i \neq j$ . Define the *grid alignment* for a point  $q = (x, y)$ , denoted  $Align(q)$  as the pair  $(i, j)$  where  $i = |\{x_i | x_i < x\}|$  and  $j = |\{y_i | y_i < y\}|$ . Then, for any two rectangles  $\langle a, b \rangle$ , and  $\langle a', b' \rangle$  we have that  $D \cap \langle a, b \rangle = D \cap \langle a', b' \rangle$  if and only if  $Align(a) = Align(a')$  and  $Align(b) = Align(b')$ . Therefore, the grid alignments of two corners of a rectangle, uniquely determine the points that lie inside that rectangle. Our protocol retrieves the points inside a rectangle by (1) determining the grid alignments corresponding to that rectangle by running the  $RL_4$  protocol (2) retrieving the points associated to those alignments by running a contiguous-PIR protocol.

The protocol to retrieve the 2-d range is presented next. If the range size (the number of retrieved points) is  $s$ , and  $k$  denotes the security parameter then its communication complexity is proportional to  $s \cdot poly(k)$ , as is the user-side computation. Server-side computation is, however, proportional to  $O(n^5)$ .

**2-d Range Retrieval Protocol**

*Range* –  $2(q_1, q_2; D)$ :

**User's input:** points  $(q_1 = (qx_1, qy_1), q_2 = (qx_2, qy_2))$

**Server's input:** database  $D = (p_1 = (x_1, y_1), \dots, p_n = (x_n, y_n))$

**User's Output:** list of all  $p_i$  such that  $x_i \in [qx_1, qx_2)$  and  $y_i \in [qy_1, qy_2)$

- Server creates auxiliary databases  $X_1, Y_1, X_2, Y_2, Z$  and  $F$  as follows:
  - $X_1, X_2 \leftarrow (x_1, x_2, \dots, x_n), Y_1, Y_2 \leftarrow (y_1, y_2, \dots, y_n)$
  - set  $t = 0$
  - for  $(i, j, k, l) \in [n]^4$ :
    - \* compute  $s$ , the number of points in  $\langle (x_i, y_j), (x_k, y_l) \rangle \cap D$
    - \*  $z_{i,j,k,l} \leftarrow (t, s)$
    - \* store the  $s$  points in  $\langle i, j \rangle \cap D$  in  $F$  at addresses  $t \dots t + s - 1$
    - \*  $t \leftarrow t + s$
- $U$  and  $S$  succinctly compute  $RL_4((qx_1, qy_1, qx_2+1, qy_2+1); X_1, Y_1, X_2, Y_2, Z)$  (see Section 3.2);  $U$  retrieves  $(t, s)$  in the end
- $U$  and  $S$  succinctly compute ContPIR (Section 2);  $U$ 's input is  $(t, s)$  and  $S$ 's is  $F$

## 5 Private $m^{th}$ -Ranked Element

Given two parties  $A$  and  $B$  with databases  $D_A$  and  $D_B$ , consider the problem of secure computation of the  $m^{th}$ -ranked element of  $D_A \cup D_B$ . In [1], Aggrawal et al. design a protocol for this problem with  $O(\log(m))$  rounds and sublinear communication complexity in the size of database. We show how to use our non-interactive private element rank protocol along with additional tricks to design a protocol with better round efficiency. Particularly, we design a protocol that requires only  $O(\log \log(m))$  rounds and has a communication complexity lower than  $O(m^\epsilon)$  for any  $0 \leq \epsilon \leq 1$ .

In what follows we assume that the output of private element rank protocol is shared between the parties (no single party learns the actual output). This can be achieved



by slightly modifying the protocol of section 3.1. More specifically, in the protocol of section 3.1, the server initially XORs all the interval labels by a random string  $r$ , and then the protocol proceeds as before. Server's share of the output will be  $r$ , while the user will receive the intended label XORed with  $r$ .

The underlying algorithm for our protocol is different from the one used by Aggrawal et al. The general idea for the algorithm is that each party searches for the  $m^{th}$ -ranked element in his/her list. One party will return the  $m^{th}$ -ranked element as a result of the search. Next we show how to perform such a search securely and in a round- and communication-efficient manner. More specifically, we design a protocol with  $O(c)$  rounds of interaction and  $O(cm^{1/c})$  communication complexity for any  $1 < c < n$ . If we let  $c = \log \log(m)$ , we achieve a protocol with  $O(\log \log(m))$  rounds and communication complexity of  $O(m^{1/\log \log(m)})$  which is smaller than  $O(m^\epsilon)$  for any  $0 \leq \epsilon \leq 1$ . It is not clear to us how to apply similar techniques to the protocol of [1] in order to achieve the same round efficiency.

Consider the first  $m$  elements of each database as our new databases. Initially, each database is divided into  $m^{1/c}$  chunks of size  $O(m^{(1-1/c)})$ . In each iteration, we determine which chunk the  $m^{th}$ -ranked element belongs to by running the element rank protocol on the database elements that separate the chunks. We then repeat the same process on the chunk that contains the  $m^{th}$ -ranked element. The number of iterations required in order to find the  $m^{th}$ -ranked element is  $c$ . Next we describe the protocol in more detail:

**Private  $m^{th}$ -ranked Element**

**Party A's input:** database  $D_A$ .  
**Party B's input:** database  $D_B$ .  
**Output:** the  $m^{th}$ -ranked element in  $D_A \cup D_B$ .

1. Assume  $D_A$  and  $D_B$  are sorted in an increasing order. Denote the set of first  $m$  elements of  $D_A$  and  $D_B$  by  $S_A$  and  $S_B$ , respectively (padded with  $+\infty$ 's to size  $m$  if necessary). For every  $1 \leq i \leq m^{1/c}$ , let  $r_A^i = r_B^i = i \times m^{(1-1/c)}$ .
2. For  $j = 1$  to  $c$ 
  - (a) For  $i = 1$  to  $m^{1/c}$ 
    - i. Parties run the private element rank protocol to compute shares of:  
 $r_A^i = Rank(S_A[r_A^i], S_B)$      $r_B^i = Rank(S_B[r_B^i], S_A)$
    - ii. Let  $R_A^i = r_A^i + r_A^i$  and  $R_B^i = r_B^i + r_B^i$ .
  - (b) Parties securely compute a circuit that takes shares of  $r_A^i, r_A^i, r_B^i,$  and  $r_B^i$  for every  $1 \leq i \leq m^{1/c}$  as inputs and outputs shares of  $O_j$  where

$$\begin{aligned}
 O_j &= S_A[r_A^i] \text{ if } R_A^i = m \\
 O_j &= S_B[r_B^i] \text{ if } R_B^i = m \\
 O_j &= 0 \quad \text{otherwise}
 \end{aligned}$$

The circuit also outputs shares of the new  $r_A^i$  and  $r_B^i$  for every  $1 \leq i \leq m^{1/c}$ , where  $r_A^i = R_A^{t_A} + i \times m^{(1-(j+1)/c)}$  and  $r_B^i = R_B^{t_B} + i \times m^{(1-(j+1)/c)}$  and  $t_A$  is the largest integer such that  $R_A^{t_A} \leq m$  (similarly for  $t_B$ ).

3. Parties securely compute a circuit that takes shares of  $O_1, \dots, O_c$  and outputs shares of  $\sum O_i$ .

## 6 One-Round PIR Is Sufficient for Private Keyword Search

We recall the setup for private keyword search (PKS) problem. The server and the client are involved. The server's input is a database  $X$  of  $n$  pairs  $(x_i, p_i)$ , where  $x_i$  is a keyword, and  $p_i$  the corresponding payload. The client's input is a searchword  $w$ . If there is a pair  $(x_i, p_i)$  in the database such that  $x_i = w$ , then the output is the corresponding payload  $p_i$ . Otherwise the output is a special symbol  $\perp$ .

In [9], Freedman et al. give a non-interactive and communication efficient solution for this functionality. Their construction requires PIR, as well as a *semantically secure homomorphic encryption scheme*. In this section we design a modified version of their scheme, which eliminates the dependency on homomorphic encryption. In fact, any one-round PIR suffices for our solution, and the communication complexity is essentially the same as that of the underlying PIR. This establishes a somewhat surprising implication:

**Theorem 3.** *If there is a one-round PIR with communication complexity  $c(n)$  then there is a one-round PKS with communication complexity  $c(n)\text{polylog}(n)$ .*

The basic idea of our solution is to replace the polynomials used in protocol of [9], by Yao's garbled circuits [15]. This already eliminates the need for homomorphic encryption which was used to encrypt the coefficients of those polynomials. Further care is necessary to make sure that after this modification the protocol remains non-interactive. We use techniques similar to those used by Cachin et al. [3] for this purpose. Due to lack of space, details of the construction are not included in this extended abstract. Please see the full version of the paper for more detail.

Finally, note that all the protocols from previous sections can be implemented from PKS only, and obviously PIR is a special case of PKS. Thus the following three problems are "equivalent up to  $\text{polylog}(n)$  factor in communication" (i.e. if any of them can be succinctly computed, then the others can too): PIR, PKS, interval labeling.

**Acknowledgment.** We would like to thank Yuval Ishai for providing us with a copy of [13], pointing to some related work and other helpful discussions.

## References

1. Aggarwal, G., Mishra, N., Pinkas, B.: Secure computation of the  $k^{\text{th}}$ -ranked element. In: Proc. of Eurocrypt (2004)
2. Aiello, B., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Proc. of Eurocrypt (2001)
3. Cachin, C., Camenisch, J., Kilian, J., Mueller, J.: One-round secure computation and secure autonomous mobile agents. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 512–523. Springer, London, UK (2000)
4. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Proc. of Eurocrypt, pp. 402–414 (1999)
5. Chang, Y.-C.: Single database private information retrieval with logarithmic communication. Cryptology ePrint Archive, Report 2004/036 (2004)

6. Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords. Technical Report TR CS0917, Department of Computer Science, Technion (1997)
7. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: Proc. of FOCS, pp. 41–50 (1995)
8. Dedić, N., Reyzin, L., Russell, S.: Unpublished manuscript
9. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudo-random functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (2005)
10. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. In: Proc. ACM STOC, ACM Press, New York (1998)
11. Goldreich, O.: Foundations of cryptography, vol. 2 (2004)
12. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proceedings of 19th Annual ACM Symposium on Theory of Computing, pp. 218–229. ACM Press, New York (1987)
13. Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, Springer, Heidelberg (to appear)
14. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: Proc. of FOCS, pp. 364–373 (1997)
15. Lindell, Y., Pinkas, B.: A proof of Yao’s protocol for secure two-party computation. Cryptology ePrint Archive, Report 2004/175 (2004)
16. Lipmaa, H.: An oblivious transfer protocol with log-squared communication. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)
17. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: pot 33rd STOC, pp. 590–599 (2001)
18. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: Proc. ACM STOC, ACM Press, New York (1999)
19. Ostrovsky, R., Skeith, W.E.: Private searching on streaming data. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 223–240. Springer, Heidelberg (2005)
20. Rabin, M.O.: How to exchange secrets by oblivious transfer. technical report tech. In: Technical Report Tech. Memo TR-81 (1981)
21. Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for NC 1. In: IEEE Symposium on Foundations of Computer Science, pp. 554–567. IEEE Computer Society Press, Los Alamitos (1999)

# Universal Algebra and Hardness Results for Constraint Satisfaction Problems<sup>\*</sup>

Benoît Larose<sup>1</sup> and Pascal Tesson<sup>2</sup>

<sup>1</sup> Department of Mathematics and Statistics, Concordia University  
larose@mathstat.concordia.ca

<sup>2</sup> Département d'Informatique et de Génie Logiciel, Université Laval  
pascal.tesson@ift.ulaval.ca

**Abstract.** We present algebraic conditions on constraint languages  $\Gamma$  that ensure the hardness of the constraint satisfaction problem  $\text{CSP}(\Gamma)$  for complexity classes L, NL, P, NP and  $\text{Mod}_p\text{L}$ . These criteria also give non-expressibility results for various restrictions of Datalog. Furthermore, we show that if  $\text{CSP}(\Gamma)$  is not first-order definable then it is L-hard. Our proofs rely on tame congruence theory and on a fine-grain analysis of the complexity of reductions used in the algebraic study of CSPs. The results pave the way for a refinement of the dichotomy conjecture stating that each  $\text{CSP}(\Gamma)$  lies in P or is NP-complete and they match the recent classification of [1] for Boolean CSP. We also infer a partial classification theorem for the complexity of  $\text{CSP}(\Gamma)$  when the associated algebra of  $\Gamma$  is the idempotent reduct of a preprimal algebra.

Constraint satisfaction problems (CSP) provide a unifying framework to study various computational problems arising naturally in artificial intelligence, combinatorial optimization, graph homomorphisms and database theory. An instance of this problem consists of a finite domain, a list of variables and constraints relating the possible values of variables: one has to decide whether the variables can be assigned values that simultaneously satisfy all constraints. This problem is of course NP-complete and so research has focused on identifying tractable subclasses of CSP. A lot of attention has been given to the case where all constraints are constructed from some *constraint language*  $\Gamma$ , i.e. some set of finitary relations over a fixed domain. In an instance of  $\text{CSP}(\Gamma)$ , all constraints are of the form  $(x_{i_1}, \dots, x_{i_k}) \in R_j \in \Gamma$ .

In their seminal work [9], Feder and Vardi conjectured that each  $\text{CSP}(\Gamma)$  either lies in P or is NP-complete. This so-called *dichotomy conjecture* is the natural extension to non-Boolean domains of a celebrated result of Schaefer [18] on the complexity of Generalized Satisfiability which states that  $\text{CSP}(\Gamma)$  is either in P or is NP-complete for any constraint language  $\Gamma$  over the Boolean domain.

Progress towards the dichotomy conjecture has been steady over the last fifteen years and has been driven by a number of complementary approaches. One

---

<sup>\*</sup> Research supported by NSERC, FQRNT and CRM. We thank Albert Atserias, Victor Dalmau, Laszlo Egri, Andrei Krokhin, Matt Valeriote and Heribert Vollmer for helpful discussions.

angle of attack relies on universal algebra: there is a natural way to associate to a set of relations  $\Gamma$  an algebra  $\mathbb{A}(\Gamma)$  whose operations are the functions that *preserve* the relations in  $\Gamma$  and one can show that the complexity of  $\text{CSP}(\Gamma)$  depends on the algebraic structure of  $\mathbb{A}(\Gamma)$ . This analysis has led to a number of key results including a verification of the dichotomy conjecture for three-element domains and the identification of wide classes of tractable CSP (see [4]).

A different, descriptive complexity approach has consisted in classifying CSPs according to the sophistication of the logical apparatus required to define the set of negative instances of  $\text{CSP}(\Gamma)$ . It was noted early on that when  $\neg\text{CSP}(\Gamma)$  is definable in the database-inspired logic programming language Datalog then  $\text{CSP}(\Gamma)$  lies in P [9] and this provides a unifying explanation for a number of (but not all) tractable cases. Further investigations have indicated strong connections between expressibility in symmetric and linear Datalog and CSPs solvable in, respectively, logarithmic space (L) and non-deterministic logspace (NL) [5][8]. From a logical perspective, the simplest class of CSPs are those which are first-order definable and recent work has provided a precise characterization for them [2][15].

From a complexity-theoretic perspective, the classification of  $\text{CSP}(\Gamma)$  as “tractable” is rather coarse. Ultimately, one would expect that every  $\text{CSP}(\Gamma)$  lying in P is in fact complete for some “fairly standard” subclass of P. Over the two-element domain, it was recently established that Schaefer’s dichotomy can in fact be refined: each  $\text{CSP}(\Gamma)$  over the Boolean domain is either **FO**-definable or is complete under  $\text{AC}^0$  isomorphisms for one of the classes L, NL,  $\oplus\text{L}$ , P or NP [1].

The present paper seeks to develop the necessary tools for a refinement of the dichotomy conjecture and for a smoother integration of the logical and algebraic approaches to the study of CSPs. As we recall in Section 2, the algebraic angle of attack relies on a number of basic reductions from  $\text{CSP}(A)$  to  $\text{CSP}(\Gamma)$  when  $\Gamma$  and  $A$  are constraint languages when the algebra  $\mathbb{A}(A)$  lies in the variety generated by  $\mathbb{A}(\Gamma)$  [3][4]. When the sole objective is to classify  $\text{CSP}(\Gamma)$  as either in P or NP-complete, polynomial-time Turing reductions are clearly good-enough. However, finer classifications require much tighter reductions and we show that all but one of them is in fact first-order. Furthermore we show that all of them preserve expressibility in Datalog and its most relevant fragments.

These reductions provide the opportunity for a systematic study of the complexity of tractable CSPs. In Section 3, we begin by proving that if  $\text{CSP}(\Gamma)$  is not first-order definable, then the problem is in fact L-hard. This result can be viewed as a first step towards more general dichotomy theorems as it exhibits a gap in the complexity of CSP. In Section 4, we use tame congruence theory and deep classification results of idempotent strictly simple algebras to obtain a number of hardness results for  $\text{CSP}(\Gamma)$ . Specifically, we consider the variety  $\mathcal{V}$  generated by  $\mathbb{A}(\Gamma)$  and give sufficient conditions on  $\mathcal{V}$  for  $\text{CSP}(\Gamma)$  to be NL-hard,  $\text{Mod}_p\text{L}$ -hard and P-hard. These also translate into necessary conditions for  $\neg\text{CSP}(\Gamma)$  to be definable in Datalog, linear Datalog and symmetric Datalog. For a given  $\Gamma$  it is possible to decide whether or not **V** fits each of these criteria [20].

In Section 5, we demonstrate the usefulness of the results by revisiting the results of [1] on CSPs over the boolean domain and by classifying the complexity of a number of  $\text{CSP}(\Gamma)$  when  $\mathbb{A}(\Gamma)$  is the idempotent reduct of a preprimal algebra.

A number of technical proofs have been omitted due to space constraints. An extended version of this abstract is available from the authors' web pages.

## 1 Preliminaries

We first set the notation and present the required basics. We refer the reader to [11] and [7] for algebraic and clone-theoretic results, to [12] for an introduction to finite model theory and descriptive complexity and to [4] for a survey on the algebraic approach to CSP.

Let  $\sigma = \{R_1, \dots, R_r, c_1, \dots, c_s\}$  be a signature, where each  $R_i$  is a relational symbol of arity  $a_i$  and each  $c_i$  is a constant symbol. A structure  $\mathbf{H}$  of signature  $\sigma$  is a tuple  $\mathbf{H} = \langle H; R_1(\mathbf{H}), \dots, R_r(\mathbf{H}), c_1(\mathbf{H}), \dots, c_s(\mathbf{H}) \rangle$  where  $H$ , the *universe of  $\mathbf{H}$* , is a non-empty set, and for each  $i$   $R_i(\mathbf{H})$  is a relation on  $H$  of arity  $a_i$ , and  $c_i(\mathbf{H})$  is some fixed element of  $H$ . We use the usual convention of using  $G, H, \dots$  to denote the universe of the structure  $\mathbf{G}, \mathbf{H}, \dots$ . Unless otherwise mentioned the signatures we deal with in this paper are purely relational (i.e. without constant symbols).

Let  $\sigma$  be a (relational) signature. Given two  $\sigma$ -structures  $\mathbf{G}$  and  $\mathbf{H}$ , a map  $f$  from  $G$  to  $H$  is a *homomorphism from  $\mathbf{G}$  to  $\mathbf{H}$*  if  $f(R_i(\mathbf{G})) \subseteq R_i(\mathbf{H})$  for all  $i$ , where for any relation  $R$  of arity  $r$  we have

$$f(R) = \{(f(x_1), \dots, f(x_r)) : (x_1, \dots, x_r) \in R\}.$$

Two structures  $\mathbf{H}$  and  $\mathbf{H}'$  are *homomorphically equivalent* if there exist homomorphisms  $\mathbf{H} \rightarrow \mathbf{H}'$  and  $\mathbf{H}' \rightarrow \mathbf{H}$ . A structure  $\mathbf{H}$  is a *core* if the only homomorphisms  $\mathbf{H} \rightarrow \mathbf{H}$  are automorphisms, or, equivalently, if it is of minimal size in its class of homomorphically equivalent structures. Every finite structure is equivalent to a structure of minimal size, and it is easy to verify that any two minimal structures are isomorphic, hence we can talk about *the* core of a structure.

Let  $\Gamma$  be a *constraint language*, i.e. a finite set of relations on the set  $H$ . Let  $\mathbf{\Gamma}$  denote a relational structure on  $H$  whose set of basic relations is  $\Gamma$  and let  $\sigma$  be its signature. We denote as  $\text{Hom}(\mathbf{\Gamma})$  the class of all finite structures of type  $\sigma$  that admit a homomorphism to  $\mathbf{\Gamma}$ . In this setting  $\mathbf{\Gamma}$  is called the *target structure*. Alternatively, we may use the notation  $\text{CSP}(\Gamma)$  for this decision problem: indeed the constraints in an instance of  $\text{CSP}(\Gamma)$  can be regarded as defining a  $\sigma$ -structure  $\mathbf{C}$  on the set of variables and a satisfying assignment is a homomorphism from  $\mathbf{C}$  to  $\mathbf{\Gamma}$ . Obviously, if  $\mathbf{\Gamma}'$  is the core of  $\mathbf{\Gamma}$  then  $\text{Hom}(\mathbf{\Gamma}) = \text{Hom}(\mathbf{\Gamma}')$ . We thus assume throughout the paper that the target structures under consideration are cores.

### 1.1 Algebras and Varieties

An  $n$ -ary operation on a set  $A$  is a map from  $A^n$  to  $A$ . The  $n$ -ary operation  $f$  on  $A$  preserves the  $k$ -ary relation  $\theta$  on  $A$  (equivalently, we say that  $\theta$  is *invariant*

under  $f$ ) if the following holds: given any matrix  $M$  of size  $k \times n$  whose columns are in  $\theta$ , applying  $f$  to the rows of  $M$  will produce a  $k$ -tuple in  $\theta$ . Given a set  $\Gamma$  of relations on  $A$ ,  $Pol(\Gamma)$  denotes the set of all operations on  $A$  that preserve all relations in  $\Gamma$ .

An algebra is a pair  $\mathbb{A} = \langle A; F \rangle$  where  $A$  is a non-empty set, called the universe of  $\mathbb{A}$ , and  $F$  is a set of operations on  $A$ , called the basic operations of  $\mathbb{A}$ . For a constraint language  $\Gamma$  over  $A$ , we denote by  $\mathbb{A}(\Gamma)$  the algebra  $\langle A; Pol(\Gamma) \rangle$ , and call it the algebra associated to  $CSP(\Gamma)$ .

The terms (polynomials) of an algebra are the operations that can be built from its basic operations (and the constants) using composition and projections. Two algebras are term (polynomially) equivalent if they have the same terms (polynomials). An operation  $f$  is *idempotent* if it satisfies  $f(x, \dots, x) = x$  for all  $x$ . The *idempotent reduct* of the algebra  $\mathbb{A}$  is the algebra with the same universe and whose basic operations are the idempotent terms of  $\mathbb{A}$ .

Subalgebras, homomorphic images and products of algebras are defined in a natural way, as for groups or rings. Technically we require the algebras to be indexed and of the same signature to define these notions, see [11]. A class of similar algebras is a *variety* if it is closed under formation of homomorphic images (H), subalgebras (S) and products (P). The *variety generated by*  $\mathbb{A}$  is denoted by  $\mathcal{V}(\mathbb{A})$ ; it is known that  $\mathcal{V}(\mathbb{A}) = HSP(\mathbb{A})$ , i.e. that every member  $\mathbb{C}$  of the variety is obtained as a homomorphic image of a subalgebra of a power of  $\mathbb{A}$ ; furthermore this power can be taken to be finite if  $\mathbb{C}$  is finite.

Tame Congruence Theory, developed by Hobby and McKenzie [11], is a powerful tool for the analysis of finite algebras. Every finite algebra has a *typeset*, which describes the local behaviour of the algebra, which contains one or more of the following 5 *types*: (1) the unary type, (2) the affine type, (3) the Boolean type, (4) the lattice type and (5) the semilattice type. There is a very tight connection between the kind of equations that are satisfied by the algebras in a variety and the types that are admitted (omitted) by a variety, i.e. those types that (do not) appear in the typeset of some algebra in the variety. The theory for idempotent algebras is somewhat more streamlined, and we now present the two results we shall require.

An algebra is *strictly simple* if it is simple (i.e., has no non-trivial congruences) and has no non-trivial subalgebras (a subalgebra is trivial if it is either the algebra itself or is 1-element.) Because a strictly simple algebra is simple it has a unique type from 1 to 5 associated to it. The next lemma (Lemma 3.1 [20]) will allow us to connect typesets of varieties to the complexity of CSP's:

**Lemma 1.1.** *Let  $\mathbb{A}$  be a finite, idempotent algebra such that  $\mathcal{V}(\mathbb{A})$  admits type  $i$ . Then there exists a strictly simple algebra of type at most  $i$  in  $HS(\mathbb{A})$  where “at most  $i$ ” refers to the ordering  $1 < 2 < 3 > 4 > 5 > 1$ .*

Szendrei has characterised all idempotent strictly simple algebras, ([19] Theorem 6.1): we need the following special cases. The 2-element set is the 2-element algebra with no basic operations  $\langle \{0, 1\}; \emptyset \rangle$ . The 2 element semilattices are the 2-element algebras with a single binary operation  $\langle \{0, 1\}; \wedge \rangle$  and  $\langle \{0, 1\}; \vee \rangle$ . The 2 element lattice is the 2 element algebra with two binary operations  $\langle \{0, 1\}; \vee, \wedge \rangle$ .

An algebra is *affine* if there is an abelian group structure on its base set such that (i)  $m(x, y, z) = x - y + z$  is a term of the algebra and (ii) every term of the algebra is affine, i.e. commutes with the operation  $m$ . Equivalently, an idempotent algebra is affine iff it is the idempotent reduct of a module.

**Lemma 1.2** ([19]). *Let  $\mathbb{A}$  be a strictly simple idempotent algebra.*

- If  $\mathbb{A}$  has unary type (type 1) then it is term equivalent to the 2-element set;
- If  $\mathbb{A}$  has affine type (type 2) then it is an affine algebra;
- if it is of semilattice (type 5) it is term equivalent to a 2 element semilattice;
- if  $\mathbb{A}$  has lattice type (type 4) it is polynomially equivalent to the 2 element lattice.

This can be used to obtain:

**Corollary 1.1.** *Let  $\mathbb{A}$  be a finite, idempotent, strictly simple algebra.*

1. If  $\mathbb{A}$  is of affine type, then there exists an Abelian group structure on the base set of  $\mathbb{A}$  such that the relation  $\{(x, y, z) : x + y = z\}$  is a subalgebra of  $\mathbb{A}^3$ ;
2. if  $\mathbb{A}$  is of semilattice type, then up to isomorphism the universe of  $\mathbb{A}$  is  $\{0, 1\}$  and the relation  $\theta = \{(x, y, z) : (y \wedge z) \rightarrow x\}$  is a subalgebra of  $\mathbb{A}^3$ ;
3. if  $\mathbb{A}$  is of lattice type, then up to isomorphism the universe of  $\mathbb{A}$  is  $\{0, 1\}$  and the relation  $\leq = \{(0, 0), (0, 1), (1, 1)\}$  is a subalgebra of  $\mathbb{A}^2$ .

## 1.2 Fragments of Datalog

Datalog was originally introduced as a database query language. We view it here simply as a means to define sets of  $\sigma$ -structures. A Datalog program over the signature  $\sigma$  consists of a finite set of *rules* of the form  $h \leftarrow b_1 \wedge \dots \wedge b_k$  where each of the  $b_i$  and  $h$  are atomic formulas of the form  $R(x_{j_1}, \dots, x_{j_r})$ . We distinguish two types of relational predicates occurring in the program: predicates  $R$  that occur at least once in the head of a rule are called *intensional database predicates* (IDBs) and are not part of  $\sigma$ . The other predicates which occur only in the body of a rule are called *extensional database predicates* and must all lie in  $\sigma$ . Precise definitions of the semantics of Datalog can be found in [14, 5, 8]: we simply illustrate the basics of the formalism with the following example.

Let  $\sigma$  be a signature consisting of a single binary relational symbol  $E$  (so that a  $\sigma$ -structure is a graph) and consider the Datalog program consisting of the rules (1)  $P(x, y) \leftarrow E(x, y)$  (2)  $P(x, y) \leftarrow P(x, z) \wedge P(z, y)$  and (3)  $G(x) \leftarrow P(x, x)$ . The program  $Q$  is providing a recursive specification of the IDB predicates  $P, G$  in terms of  $E, P$  and  $G$ . The predicate  $P$  is intended to include  $(x, y)$  if there is a path from  $x$  to  $y$ . The first rule states that this holds if  $(x, y)$  is an edge and the second that, recursively, this holds if there is a path from  $x$  to some  $z$  and from  $z$  to  $y$ . The predicate  $G$  then contains the  $x$  such that there is a directed cycle around  $x$ . One of the IDBs of the Datalog program is chosen as a target and we say that a  $\sigma$ -structure is accepted by the program if that target IDB is non-empty. The program above with  $G$  as its goal thus defines the set of graphs with a directed cycle.



Rules which contain only EDBs in their body (such as (1) above) are called *non-recursive rules* and those containing at most one IDB in their body (such as (1) and (3)) are *linear*. Although the above example contains the non-linear rule (2), it is easy to see that an equivalent linear program could be obtained by replacing (2) with  $P(x, y) \leftarrow P(x, z) \wedge E(z, y)$ . A program is said to be symmetric if it is linear and such that each recursive rule  $\mathcal{R}$  is accompanied by its symmetric  $\mathcal{R}^r$ , where  $\mathcal{R}^r$  is obtained by exchanging from  $\mathcal{R}$  by exchanging the roles of the IDBs in its head and body. The symmetric of the above rule would be  $P(x, z) \leftarrow P(x, y) \wedge E(z, y)$ .

The expressive power of Datalog and its linear, symmetric and non-recursive fragments have been important tools in the study of CSP. A very nice result of [2] shows that  $\text{CSP}(\Gamma)$  is definable by a first-order sentence iff  $\neg\text{CSP}(\Gamma)$  is definable by a non-recursive Datalog program and consequently all such  $\text{CSP}(\Gamma)$  are solvable in co-NLOGTIME. Moreover, expressibility of  $\neg\text{CSP}(\Gamma)$  in symmetric, linear and general Datalog is a sufficient (and wide-encompassing) condition for  $\text{CSP}(\Gamma)$  to lie in respectively L [8], NL [5] and P [9].

## 2 Nature of the Algebraic and Clone-Theoretic Reductions

The following theorem is our starting point for a fine-grained analysis of the complexity of constraint satisfaction problems. A relation  $\theta$  is *irredundant* if for each two distinct coordinates  $i$  and  $j$  there exists a tuple  $\bar{x}$  of  $\theta$  with  $x_i \neq x_j$ .

**Theorem 2.1.** *Let  $\Gamma$  be a finite set of relations on  $A$  such that  $\Gamma$  is a core. Let  $\mathbb{A}$  denote the idempotent reduct of the algebra associated to  $\Gamma$ .*

1. *Let  $\mathbb{C}$  be a finite algebra in  $\mathcal{V}(\mathbb{A})$ , and let  $\Gamma_0$  be a finite set of relations invariant under the basic operations of  $\mathbb{C}$ . Then there exists a logspace many-one reduction of  $\text{CSP}(\Gamma_0)$  to  $\text{CSP}(\Gamma)$ . Furthermore, if  $\neg\text{CSP}(\Gamma)$  is expressible in (linear, symmetric) Datalog, then so is  $\neg\text{CSP}(\Gamma_0)$ .*
2. *If furthermore  $\mathbb{C} \in \text{HS}(\mathbb{A})$  and the relations in  $\Gamma_0$  are irredundant, then the above reduction is first-order.*

The proof, although not conceptually difficult, is technical and is omitted for space. The constructions are given for ten basic reductions which can be composed to obtain the two claims above: their principles are not new [13,3] although most were never explicitly shown to be first order or to preserve expressibility in the linear and symmetric fragments of Datalog (the case of Datalog is treated in [17]). It should be noted that logspace reductions are the best we can hope for in the first half of the statement: indeed, it is straightforward from the definitions to see that if  $\Gamma_0 = \Gamma \cup \{=\}$  then one has  $\mathbb{A}(\Gamma) = \mathbb{A}(\Gamma_0)$ . But if  $\text{CSP}(\Gamma)$  is first-order definable then  $\text{CSP}(\Gamma_0)$  is L-complete (see e.g. [8]) and so there can be no first-order reduction from  $\text{CSP}(\Gamma_0)$  to  $\text{CSP}(\Gamma)$ .

### 3 CSP's That Are Not FO Are L-Hard

In this section we show that for every finite set  $\Gamma$ , if  $CSP(\Gamma)$  is not first-order expressible then it is L-hard. We require a characterisation of first-order definable CSP's from [15]. Consider the signature  $\sigma = \{R_1, \dots, R_r\}$  where  $R_i$  is a relational symbol of arity  $a_i$ . For an integer  $n$  the  $n$ -link of type  $\sigma$  is the  $\sigma$ -structure

$$\mathbf{L}_n = \langle \{0, 1, \dots, n\}; R_1(\mathbf{L}_n), \dots, R_r(\mathbf{L}_n) \rangle,$$

such that  $R_i(\mathbf{L}_n) = \cup_{j=1}^n \{j-1, j\}^{a_i}$  for  $i = 1, \dots, r$ . Intuitively, a link is obtained from a path  $0, 1, \dots, n$  by replacing each edge by the relational structure of type  $\sigma$  on 2 elements whose basic relations are of maximal size.

Let  $\mathbf{A}$  and  $\mathbf{B}$  be two  $\sigma$ -structures. The  $\mathbf{A}$ -th power of  $\mathbf{B}$  is the  $\sigma$ -structure

$$\mathbf{B}^{\mathbf{A}} = \langle B^{\mathbf{A}}; R_1(\mathbf{B}^{\mathbf{A}}), \dots, R_r(\mathbf{B}^{\mathbf{A}}) \rangle,$$

where  $B^{\mathbf{A}}$  is the set of all maps from  $A$  to  $B$ , and for  $i = 1, \dots, r$  the relation  $R_i(\mathbf{B}^{\mathbf{A}})$  consists of all tuples  $(f_1, \dots, f_{a_i})$  such that  $(f_1(x_1), \dots, f_{a_i}(x_{a_i})) \in R_i(\mathbf{B})$  whenever  $(x_1, \dots, x_{a_i}) \in R_i(\mathbf{A})$ .

Let  $\pi_1$  and  $\pi_2$  denote the two projections from  $A^2$  to  $A$ .

**Lemma 3.1** ([15]). *Let  $\Gamma$  be a finite set of relation on  $A$  such that  $\Gamma$  is a core. Then  $CSP(\Gamma)$  is first-order definable if and only if for some  $n$  there exists a homomorphism  $P : \mathbf{L}_n \rightarrow \Gamma(\Gamma^2)$  such that  $P(0) = \pi_1$  and  $P(n) = \pi_2$ .*

**Theorem 3.1.** *For any finite constraint language  $\Gamma$  the problem  $CSP(\Gamma)$  is either first-order definable or is L-hard.*

*Proof (Sketch).* We assume that  $\Gamma$  is a core. Let  $\sigma = \{R_1, \dots, R_r\}$  be the signature of the structure  $\Gamma$  and let  $\Gamma' = \Gamma \cup \{\{a\} : a \in A\}$ . For  $a \in A$  let  $S_a$  be a relational symbol for  $\{a\}$ , and let  $\sigma' = \{R_1, \dots, R_r\} \cup \{S_a : a \in A\}$  be the signature of the structure  $\Gamma'$ . It suffices to show that if  $CSP(\Gamma)$  is not first-order definable then  $CSP(\Gamma')$  is L-hard (see full version): for this we construct a first-order reduction of NOT  $st$ -connectivity (which is L-hard under first-order reductions [12]) to  $CSP(\Gamma')$ . Consider the vocabulary of graphs with two specified vertices,  $\tau = \{E, s, t\}$  where  $E$  is a binary relational symbol and  $s$  and  $t$  are constant symbols. We shall describe a first-order interpretation  $\mathcal{I}$  of  $\sigma'$  into  $\tau$  assigning to each graph  $\mathbf{G}$  with distinguished vertices  $s$  and  $t$  a structure  $\mathbf{K} = \mathcal{I}(\mathbf{G})$  of type  $\sigma'$  such that  $\mathbf{K}$  admits a homomorphism to  $\Gamma'$  precisely when  $s$  and  $t$  are not connected in  $\mathbf{G}$ .

It is clear that the following defines a symmetric relation  $\sim$  on  $\Gamma(\Gamma^2)$ : let  $g \sim h$  if there exists a homomorphism  $F : \mathbf{L}_1 \rightarrow \Gamma(\Gamma^2)$  such that  $F(0) = g$  and  $F(1) = h$ . It is also clear by definition of the links that Lemma 3.1 shows this: for a core  $\Gamma$ ,  $CSP(\Gamma)$  is first-order definable precisely when the projections are connected in the graph defined by  $\sim$ .

We simply describe the reduction: the argument that it is indeed first-order is omitted. Given a graph  $\mathbf{G}$  with specified vertices  $s$  and  $t$ , we construct a  $\sigma$ -structure  $\mathbf{H}$  which is obtained from  $\mathbf{G}$  by replacing each edge by the link  $\mathbf{L}_1$  (in

the same manner that links are obtained from paths). Consider now the product  $\sigma$ -structure  $\mathbf{H} \times \Gamma^2$ , that we transform into the  $\sigma'$ -structure  $\mathbf{K} = \mathcal{I}(\mathbf{G})$  as follows: for each  $a \in A$  we define  $S_a(\mathbf{K})$  to contain all elements  $(g, c, d)$  such that  $g = s$  and  $c = a$  OR  $g = t$  and  $d = a$ . We first show that the above is indeed a reduction of NOT  $st$ -connectivity to  $CSP(\Gamma')$ . Suppose there is a homomorphism  $f$  from  $\mathbf{K}$  to  $\Gamma'$ : in particular it is a homomorphism of  $\sigma$ -structures  $f : \mathbf{H} \times \Gamma^2 \rightarrow \Gamma$ , which, by the natural property of products, induces a homomorphism  $F$  from  $\mathbf{H}$  to  $\Gamma^{\Gamma^2}$ ; by definition of the relations  $S_a(\mathbf{K})$ , it is easy to verify that  $F(s) = \pi_1$  and  $F(t) = \pi_2$ . Indeed, we have  $F(s)(c, d) = f(s, c, d) = c$  and  $F(t)(c, d) = f(t, c, d) = d$ . Since  $CSP(\Gamma)$  is not first-order definable, there is no path in  $\Gamma$  between the projections, hence there cannot be a path in  $\mathbf{G}$  from  $s$  to  $t$ . Conversely, suppose there is no such path in  $\mathbf{G}$ . Then define a map  $f$  from  $\mathbf{H} \times \Gamma^2$  to  $\Gamma$  by setting  $f(g, c, d) = c$  if there is a path in  $\mathbf{G}$  from  $s$  to  $g$  and  $f(g, c, d) = d$  otherwise. This is clearly well-defined, and obviously preserves all the relations  $S_a$ . It is easy to see that  $f$  also preserves all relations  $R_i$ : indeed, the map  $F : \mathbf{H} \rightarrow \Gamma^{\Gamma^2}$  induced by  $f$  maps all elements to one of two projections which are “loops” in any power structure.

### 4 Main Theorems

We present our two main theorems. The first provides sufficient algebraic criteria for the hardness of  $CSP(\Gamma)$  for a number of natural complexity classes. The second expresses these same lower bounds in descriptive complexity terms.

**Theorem 4.1.** *Let  $\Gamma$  be a finite set of relations on  $A$  such that  $\Gamma$  is a core, and let  $\mathbb{A} = \mathbb{A}(\Gamma)$ . Then:*

1. *If  $\mathcal{V}(\mathbb{A})$  admits the unary type, then  $CSP(\Gamma)$  is NP-complete under FO reductions.*
2. *If  $\mathcal{V}(\mathbb{A})$  omits the unary type but admits the affine type, then there exists a prime  $p$  such that  $CSP(\Gamma)$  is  $Mod_p L$ -hard under FO reductions.*
3. *If  $CSP(\Gamma)$  is not FO, then it is L-hard under FO reductions.*
4. *If  $\mathcal{V}(\mathbb{A})$  omits the unary, and semilattice types, but admits the lattice type, then  $CSP(\Gamma)$  is NL-hard under FO reductions.*
5. *If  $\mathcal{V}(\mathbb{A})$  omits the unary type, but admits the semilattice type, then  $CSP(\Gamma)$  is P-hard under FO reductions.*

*Proof (Sketch).* (3) is the content of Theorem 3.1, and (1) follows from a result of [3]: the reduction there is actually first-order by Theorem 2.1.

It follows from results in [11] that if  $\mathbb{A}$  satisfies the hypothesis of one of (2), (4) or (5) then so does its idempotent reduct, which we denote by  $\mathbb{B}$ . By Lemma 1.1, if  $\mathbb{B}$  satisfies the hypothesis (i) then there exists a strictly simple algebra  $\mathbb{C} \in HS(\mathbb{A})$  of type (i). By Corollary 1.1 it means that, in case (2), there exists an Abelian group structure on the base set of  $\mathbb{B}$  such that the 3-ary relation  $\mu = \{(x, y, z) : x + y = z\}$  is invariant under the operations of  $\mathbb{B}$ . Consider the set  $\Gamma'$  that consists of the relation  $\mu$ , the unary relation  $B = \{b\}$  where  $b$  is some

non-zero element of  $\mathbb{B}$  such that  $pb = 0$  for some prime  $p$ , and the unary relation  $Z = \{0\}$ . We show in the full version of the paper how Theorem 2.1 can be used to obtain a first-order reduction of  $\text{CSP}(\Gamma')$  to  $\text{CSP}(\Gamma)$  for some  $\Gamma$  for which  $\text{CSP}(\Gamma)$  is mod- $p$  L hard under FO reductions.

We proceed as in the other two cases. In case (4), we find an FO reduction of  $\text{CSP}(\Gamma')$  to  $\text{CSP}(\Gamma)$  where  $\Gamma' = \{\leq, \{0\}, \{1\}\}$ . There is a straightforward FO-reduction from the directed graph unreachability problem to  $\text{CSP}(\Gamma')$  and the former problem is NL-complete under first-order reductions [12].

In case (5) we find a first-order reduction of  $\text{CSP}(\Gamma')$  to  $\text{CSP}(\Gamma)$  where  $\Gamma' = \{\theta, \{0\}, \{1\}\}$  where  $\theta = \{(x, y, z) : (y \wedge z) \rightarrow x\}$ . It is again straightforward to show that  $\text{CSP}(\Gamma')$  admits a natural FO reduction from HORN-3-SAT which is P-hard under FO-reductions [12].

Similar arguments lead to an analogous result in which hardness is replaced by non-expressibility for fragments of Datalog. Note that (1) is shown in [17] while (2) and (3) rely on the fact that  $\neg\text{CSP}(\{\leq, \{0\}, \{1\}\})$  and  $\neg\text{CSP}(\{\theta, \{0\}, \{1\}\})$  are not definable in, respectively, symmetric and linear Datalog [6].

**Theorem 4.2.** *Let  $\Gamma$  be a finite set of relations on  $A$  such that  $\Gamma$  is a core, and let  $\mathbb{A} = \mathbb{A}(\Gamma)$ . Then:*

1. *If  $\mathcal{V}(\mathbb{A})$  admits the unary or affine types, then  $\neg\text{CSP}(\Gamma)$  is not in Datalog.*
2. *If  $\mathcal{V}(\mathbb{A})$  omits the unary type, but admits the semilattice type, then  $\neg\text{CSP}(\Gamma)$  is not in linear Datalog.*
3. *If  $\mathcal{V}(\mathbb{A})$  omits the unary and semilattice types, but admits the lattice type, then  $\neg\text{CSP}(\Gamma)$  is not in symmetric Datalog.*

## 5 Applications

Because the algebraic criteria used in Theorems 4.1 and 4.2 are all decidable [20], they are a very convenient first step when studying the complexity of  $\text{CSP}(\Gamma)$  for a specific  $\Gamma$  or a specific class of them. We first show that our criteria match Allender et al.'s [1] description of the complexity of Boolean CSP's and line up exactly with the expressibility in restrictions of Datalog. We finally use them to study CSPs linked to preprimal algebras.

### 5.1 Boolean CSP's

**Theorem 5.1.** *Let  $\Gamma$  be a set of relations on  $\{0, 1\}$  such that  $\Gamma$  is a core. Let  $\mathcal{V}$  denote the variety generated by  $\mathbb{A}(\Gamma)$ .*

1. *If  $\mathcal{V}$  admits the unary type then  $\text{CSP}(\Gamma)$  is NP-complete, and  $\neg\text{CSP}(\Gamma)$  is not expressible in Datalog.*
2. *If  $\mathcal{V}$  omits the unary type but admits the affine type, then  $\text{CSP}(\Gamma)$  is  $\oplus L$ -complete and  $\neg\text{CSP}(\Gamma)$  is not expressible in Datalog.*
3. *if  $\mathcal{V}$  admits only the Boolean type,  $\text{CSP}(\Gamma)$  is either first-order definable or L-complete; if  $\Gamma$  is finite,  $\neg\text{CSP}(\Gamma)$  is expressible in symmetric Datalog;*

4. if  $\mathcal{V}$  omits the unary, affine and semilattice types, but admits the lattice type, then  $CSP(\Gamma)$  is NL-complete; if  $\Gamma$  is finite, then  $\neg CSP(\Gamma)$  is expressible in linear Datalog, but not in symmetric Datalog;
5. if  $\mathcal{V}$  omits the unary and affine types, but admits the semilattice type, then  $CSP(\Gamma)$  is P-complete; if  $\Gamma$  is finite then  $\neg CSP(\Gamma)$  is expressible in Datalog, but not in linear Datalog.

## 5.2 Preprimal Algebras

We now use our results to investigate the descriptive and computational complexity of CSP's whose associated algebra is the idempotent reduct of a preprimal algebra. A finite algebra  $\mathbb{A}$  is *preprimal* if its clone of term operations is maximal in the lattice of clones, i.e. is properly contained in the set of all operations on the base set  $A$  but there is no clone strictly between these. Maximal clones satisfy remarkable properties, for instance every clone is contained in some maximal clone and they are finite in number. They were completely classified by I.G. Rosenberg (see [7]), thereby furnishing an explicit criterion to determine if a set of operations generates all operations on a finite set  $A$  by composition. Alternatively, one may view CSP's whose algebra is preprimal as those whose constraint language is minimal, in the sense that it is non-trivial but every of its constraints can be inferred from any other non-trivial constraint in the language. It is easy to see that any maximal clone may be expressed in the form  $Pol(\theta)$  for some relation  $\theta$ ; we shall investigate problems  $CSP(\Gamma)$  where  $Pol(\Gamma) = Pol(\{\theta\} \cup \{\{a\} : a \in A\})$ , i.e. such that the associated algebra of the problem  $CSP(\Gamma)$  is the idempotent reduct of a preprimal algebra. We follow Rosenberg's classification of the relations  $\theta$  that yield maximal clones, see pages 230-231 of [7]. We also require an effective characterisation of FO definable CSP's from [15]. Let  $\mathbf{G}$  be a relational structure and let  $a, b \in G$ . We say that  $b$  *dominates*  $a$  in  $\mathbf{G}$  if for any basic relation  $R$  of  $\mathbf{G}$ , and any tuple  $t \in R$ , replacement of any occurrence of  $a$  by  $b$  in  $t$  will yield a tuple of  $R$ . If  $\mathbf{\Gamma}$  is a relational structure on  $A$ , we say that the structure  $\mathbf{\Gamma}^2$  *dismantles to the diagonal* if one may obtain, by successive removals of dominated elements of  $\mathbf{\Gamma}^2$ , the diagonal  $\{(a, a) : a \in A\}$ .

**Lemma 5.1** ([15]). *Let  $\Gamma$  be a set of relations such that  $\mathbf{\Gamma}$  is a core. Then  $CSP(\Gamma)$  is first-order expressible if and only if  $\mathbf{\Gamma}^2$  dismantles to the diagonal.*

(P) (Permutation) Here  $\theta = \pi^\circ$  for some permutation  $\pi$  which is fixed point free and of prime order. In this case  $\neg CSP(\Gamma)$  is expressible in symmetric Datalog by [8]. In particular  $CSP(\Gamma)$  is in L and in fact is L-complete: it is easy to show that  $\mathbf{\Gamma}^2$  does not dismantle to the diagonal and thus that  $CSP(\Gamma)$  is not FO-definable and L-hard by Theorem 3.1.

(E) (Equivalence) Here  $\theta$  is a non-trivial equivalence relation on  $A$ ; following [8] the problem  $\neg CSP(\Gamma)$  is expressible in symmetric Datalog. Hence  $CSP(\Gamma)$  is in L, and again L-complete because one can also show that  $\mathbf{\Gamma}^2$  does not dismantle to the diagonal.

(A) (Affine) In this case  $\theta = \{(a, b, c, d) : a + b = c + d\}$  where  $\langle A; +, 0 \rangle$  is some Abelian  $p$ -group for some prime  $p$ . Notice that the associated algebra is

affine (in the sense defined earlier) and so the variety it generates admits the affine type, and hence by Theorems 4.2 and 4.1  $CSP(\Gamma)$  is not in Datalog, and it is  $Mod_pL$ -hard and in fact  $Mod_pL$ -complete (see full version).

(C) (Central) Here  $\theta$  is a  $k$ -ary relation ( $k \geq 1$ ) different from  $A^k$  that must (among other things) have a *central* element, i.e. there is some  $c \in A$  such that  $\theta$  contains every tuple with an occurrence of  $c$ . In that case  $\Gamma^2$  does dismantles to the diagonal and  $CSP(\Gamma)$  is FO-definable. It follows from Theorem 5 of [8] that  $\neg CSP(\Gamma)$  is in symmetric Datalog, and if  $\Gamma$  does not contain a so-called *biredundant relation* then  $CSP(\Gamma)$  is actually first-order definable [15].

(R) (Regular) Here  $\theta$  is a  $k$ -ary ( $k \geq 3$ ) *regular* relation defined as follows. Let  $\mathbf{S}$  denote the structure with universe  $\{1, \dots, k\}$  and one basic relation  $\theta(\mathbf{S})$  of arity  $k$ , consisting of all tuples  $(x_1, \dots, x_k)$  with at least one repeated coordinate. Operations that preserve this relation are known to be the following: all non-surjective operations and all essentially unary operations i.e. that depend on only one variable [7]. In particular, no non-trivial idempotent operation preserves  $\theta(\mathbf{S})$ . For any positive integer  $m$  let  $\mathbf{S}^m$  denote the  $m$ -th power of this structure. A  $k$ -ary relation on the set  $A$  is *regular* if there exists some positive integer  $m$ , and a surjective map  $\mu$  from  $A$  to  $\mathbf{S}^m$  such that  $\theta = \mu^{-1}(\theta(\mathbf{S}^m))$ . Clearly, in this case, the structure  $\langle A; \theta \rangle$  retracts onto  $\mathbf{S}^m$ , and it is easy to see that  $\mathbf{S}^m$  retracts onto  $\mathbf{S}$  ( $\mathbf{S}$  embeds in  $\mathbf{S}^m$  via the map  $x \mapsto (x, 1, \dots, 1)$ .) From results in [17], the relation  $\theta$  cannot be invariant under a so-called Taylor operation, and thus  $\mathbb{A}(\Gamma)$  generates a variety that admits the unary type and  $CSP(\Gamma)$  is NP-complete.

(O) (Order) In this last case,  $\theta$  is a bounded order relation, i.e. a reflexive, antisymmetric, transitive relation  $\leq$  with elements 0 and 1 such that  $0 \leq x \leq 1$  for all  $x \in A$ . In that case the variety generated by the associated algebra admits type 4 or 5 and hence by Theorems 4.2 and 4.1  $\neg CSP(\Gamma)$  is not expressible in symmetric Datalog, and  $CSP(\Gamma)$  is NL-hard. There is not much that is known at this time on these CSP's, either from the algebraic or the complexity point of view: the class of so-called order-primal algebras is vast and quite complex. There are posets for which the problem is NP-complete, others for which the problem is NL-complete: the examples known to be in *NL* have their complement definable in linear Datalog. It is also possible to construct, from a non-bounded example found in [17], a bounded poset whose associated problem is tractable and mod- $p$  L hard. Similarly, from an example found in [16], one may construct a tractable example whose variety admits type 5 and hence is P-complete.

Our case analysis can be summarized as follows.

**Theorem 5.2.** *Let  $\Gamma$  be a finite set of relations such that  $Pol(\Gamma) = Pol(\{\theta\} \cup \{\{a\} : a \in A\})$  where  $Pol(\theta)$  is a maximal clone. If the maximal clone is of type (P), (E), (A), (C), (R), (O), then  $CSP(\Gamma)$  satisfies the properties given in the following:*

- (P) (Permutation) Symmetric Datalog; L-complete.
- (E) (Equivalence) Symmetric Datalog; L-complete.
- (A) (Affine) not Datalog;  $Mod_pL$ -complete for some prime  $p$ .
- (C) (Central) Symmetric Datalog; first-order definable if  $\Gamma$  contains no biredundant relation, L-complete otherwise.

**(R)** (Regular) NP-complete.

**(O)** (Order) not in symmetric Datalog; NL-hard; some cases are known to be NP-complete, some known to be NL-complete, some P-complete.

## References

1. Allender, E., Bauland, M., Immerman, N., Schnoor, H., Vollmer, H.: The complexity of satisfiability problems: Refining Schaefer's theorem. In: Jędrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 71–82. Springer, Heidelberg (2005)
2. Atserias, A.: On digraph coloring problems and treewidth duality. In: Proc. 21st Conf. on Logic in Comp. Sci. (LICS '05), pp. 106–115 (2005)
3. Bulatov, A., Krokhin, A., Jeavons, P.: Constraint satisfaction problems and finite algebras. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 272–282. Springer, Heidelberg (2000)
4. Cohen, D., Jeavons, P.G.: The complexity of constraint languages. In: Handbook of Constraint Programming (2006)
5. Dalmau, V.: Linear Datalog and bounded path duality of relational structures. Logical Methods in Computer Science 1(1) (2005)
6. Dalmau, V., Egri, L., Larose, B., Tesson, P.: On the limits of expressivity of linear and symmetric Datalog. Document in preparation (2007)
7. Denecke, K., Wismath, S.L.: Universal Algebra and Applications in Theoretical Computer Science. CRC/C&H, Boca Raton (2002)
8. Egri, L., Larose, B., Tesson, P.: Symmetric datalog and constraint satisfaction problems in logspace (Submitted, 2007)
9. Feder, T., Vardi, M.Y.: The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. SIAM J. on Computing 28(1), 57–104 (1999)
10. Feder, T., Vardi, M.Y.: Homomorphism closed vs. existential positive. In: Feder, T., Vardi, M.Y. (eds.) Proc. 18th Symp. on Logic in Comp. Sci (LICS 2003), pp. 311–320 (2003)
11. Hobby, D., McKenzie, R.: The Structure of Finite Algebras. Contemporary Mathematics, vol. 76. American Mathematical Society, New York (1988)
12. Immerman, N.: Descriptive Complexity. Graduate Texts in Computer Science. Springer, Heidelberg (1999)
13. Jeavons, P.: On the algebraic structure of combinatorial problems. Theor. Comput. Sci. 200(1-2), 185–204 (1998)
14. Kolaitis, P.G., Vardi, M.Y.: On the expressive power of datalog: Tools and a case study. J. Comput. Syst. Sci. 51(1), 110–134 (1995)
15. Larose, B., Loten, C., Tardif, C.: A characterization of first-order constraint satisfaction problems. In: Proc. 21st Symp. on Logic in Comp, pp. 201–210 (2006)
16. Larose, B., Zádori, L.: Finite posets and topological spaces in locally finite varieties. Algebra Universalis 52(2-3), 119–136 (2005)
17. Larose, B., Zádori, L.: Bounded width problems and algebras. Algebra Universalis (2006)
18. Schaefer, T.J.: The complexity of satisfiability problems. In: Proc. 10<sup>th</sup> ACM STOC, pp. 216–226. ACM Press, New York (1978)
19. Szendrei, A.: A survey on strictly simple algebras and minimal varieties. Research and Exposition in Mathematics, pp. 209–239. Heldermann Verlag (1992)
20. Valeriote, M.: A subalgebra intersection property for congruence distributive varieties. In: Canadian J. of Math. (to appear)

# On the Power of $k$ -Consistency

Albert Atserias<sup>1</sup>, Andrei Bulatov<sup>2</sup>, and Victor Dalmau<sup>3</sup>

<sup>1</sup> Universitat Politècnica de Catalunya, Barcelona, Spain

atserias@lsi.upc.edu

<sup>2</sup> Simon Fraser University, Vancouver, Canada

abulatov@cs.sfu.ca

<sup>3</sup> Universitat Pompeu Fabra, Barcelona, Spain

victor.dalmau@upf.edu

**Abstract.** The  $k$ -consistency algorithm for constraint-satisfaction problems proceeds, roughly, by finding all partial solutions on at most  $k$  variables and iteratively deleting those that cannot be extended to a partial solution by one more variable. It is known that if the core of the structure encoding the scopes of the constraints has treewidth at most  $k$ , then the  $k$ -consistency algorithm is always correct. We prove the exact converse to this: if the core of the structure encoding the scopes of the constraints does not have treewidth at most  $k$ , then the  $k$ -consistency algorithm is not always correct. This characterizes the exact power of the  $k$ -consistency algorithm in structural terms.

## 1 Introduction

Let  $\mathbf{A}$  and  $\mathbf{B}$  be two relational structures of the same type. For concreteness, we can think of  $\mathbf{A}$  and  $\mathbf{B}$  as directed graphs, each consisting of a set of vertices and a binary relation on the vertices. A homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$  is a map from the domain of  $\mathbf{A}$  to the domain of  $\mathbf{B}$  that preserves all the relations. Homomorphisms play a prominent role in combinatorics, logic, and algebra, and also in computer science. Consider for example the constraint-satisfaction problem, where we are given a set of variables that range over a domain of values, and a set of constraints between tuples of variables and tuples of values. The goal is to find an assignment of values to the variables in such a way that all given constraints are fulfilled. It was observed by Feder and Vardi [9] that this problem can be phrased as a homomorphism question between a relational structure  $\mathbf{A}$  encoding the set of constraint variables (scopes), and a relational structure  $\mathbf{B}$  encoding the set of valid assignment to those variables.

The  $k$ -consistency algorithm is a well-known heuristic algorithm to decide the existence of a homomorphism between two structures, or equivalently, to solve constraint-satisfaction problems. In order to simplify the exposition, let us focus again on finite directed graphs  $\mathbf{A} = (A, E^{\mathbf{A}})$  and  $\mathbf{B} = (B, E^{\mathbf{B}})$  and let us fix  $k = 1$ . The 1-consistency algorithm is commonly referred to as the *arc-consistency algorithm*. This algorithm proceeds in rounds by iteratively reducing the set of possible places  $L(a) \subseteq B$  where a vertex  $a \in A$  may be mapped.



Initially, every  $a \in A$  can be mapped to any  $b \in B$ , so we start with  $L(a) = B$ . At each round, if there exists an arc  $(a, a') \in E^{\mathbf{A}}$  and a  $b \in L(a)$  for which no  $b' \in L(a')$  exists such that  $(b, b') \in E^{\mathbf{B}}$ , we remove  $b$  from  $L(a)$ . Similarly, if there exists a  $b' \in L(a')$  for which no  $b \in L(a)$  exists such that  $(b, b') \in E^{\mathbf{B}}$ , we remove  $b'$  from  $L(a')$ . This process is repeated until there are no more changes in the  $L(a)$ 's. If at termination  $L(a)$  is empty for some  $a \in A$ , we can guarantee that there exists no homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ . Otherwise, we say that the instance  $\mathbf{A}, \mathbf{B}$  passes the arc-consistency test. In this case we know that if there exists a homomorphism  $h : \mathbf{A} \rightarrow \mathbf{B}$ , we must have  $h(a) \in L(a)$  for every  $a \in A$ . Henceforth, the arc-consistency algorithm can be used in order to narrow the possible space of solutions, and indeed, many of the practical CSP solvers use some form of consistency in order to prune the search tree. Furthermore, most of the known tractable subcases of the CSP are solvable by testing some sort of consistency.

The  $k$ -consistency test for general  $k$  is the natural generalization of this algorithm to  $k$ -tuples. The main goal of this paper is to study the power of the  $k$ -consistency test as a tool to decide the existence of a solution by itself. More precisely, we are interested in characterizing under which circumstances we can guarantee that every instance passing the  $k$ -consistency test has a solution.

Note, first, that the consistency test runs in time polynomial in  $|A| \cdot |B|$ , which is polynomial in the size of the input. Therefore, since the general homomorphism problem is NP-complete, we cannot expect it to be correct on every instance. Interestingly, though, it is known that the algorithm is correct when the underlying graph of  $\mathbf{A}$  is acyclic [10]. This gives a large class of inputs where the algorithm can be used to find homomorphisms in polynomial time. It was later observed that it suffices if the core of  $\mathbf{A}$  is acyclic [6], where the core of a relational structure  $\mathbf{A}$  is the smallest substructure that has homomorphisms from and to  $\mathbf{A}$ . It is known that such a substructure exists and is unique up to isomorphism [13]. This widens the class of instances where the algorithm works correctly even further. But is that all?

*Main result.* The main result of this paper is the complete answer to the question above. In fact, our result answers the corresponding question for the  $k$ -consistency test. In this context, the role of graph acyclicity is played by the concept of treewidth, which is a measure to calibrate the similarity of a graph with a tree.

Treewidth was introduced in the deep work on graph minors by Robertson and Seymour, and has played an important role in algorithmic graph theory since then. For constraint-satisfaction problems, treewidth was identified as useful by Freuder [11], and later revisited by several others [9, 19, 14, 6]. Freuder observed that if the treewidth of  $\mathbf{A}$  is at most  $k$ , then the  $k$ -consistency algorithm is always correct. As with the acyclic case, it was later observed that it suffices that the treewidth of the core of  $\mathbf{A}$  be bounded by  $k$ . Thus, it was proved in [6] that if the treewidth of the core of  $\mathbf{A}$  is at most  $k$ , then the  $k$ -consistency algorithm is always correct. Our main result is an exact converse to this: if the

treewidth of the core of  $\mathbf{A}$  is more than  $k$ , then the  $k$ -consistency algorithm is not always correct. Note that since treewidth at most 1 agrees with acyclicity of the underlying undirected graph, our main result implies, in particular, that if the core of  $\mathbf{A}$  is not acyclic, then the arc-consistency test is not always correct.

*Related work.* The notion of  $k$ -consistency has proven to be very robust and, besides being one of the central concepts in theory of constraint-satisfaction problems, has also emerged independently in areas as diverse as finite model theory [18], graph theory [16], and proof complexity [1].

The limits of the  $k$ -consistency algorithm as a method for finding homomorphisms had been studied before to some extent. First, for each fixed  $k$ , concrete examples where the algorithm is not correct can be easily found. For example, let  $\mathbf{A}$  be a complete graph on  $k + 2$  vertices, and let  $\mathbf{B}$  be a complete graph on  $k + 1$  vertices. It is not hard to see that there is no homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$  yet this instance passes the  $k$ -consistency test.

Second, Feder and Vardi [9] proved that there exists a fixed finite structure  $\mathbf{B}$  for which it is possible to determine the existence of a homomorphism  $\mathbf{A} \rightarrow \mathbf{B}$  in polynomial time, yet the  $k$ -consistency algorithm fails for every fixed  $k$ . In fact, the structure  $\mathbf{B}$  is very explicit and corresponds to the constraint-satisfaction problem of solving systems of linear equations over the two-element field.

Third, Grohe [12] proved the following very general result. Let  $\mathcal{F}$  be a class of structures and consider the restricted homomorphism problem when  $\mathbf{A}$  is taken from  $\mathcal{F}$  and  $\mathbf{B}$  is an arbitrary structure. For which  $\mathcal{F}$ 's is this problem solvable in polynomial time? We know already from [6] that if the class of cores of  $\mathcal{F}$  has bounded treewidth, then the problem is solvable in polynomial time. Assuming a conjecture in parameterized complexity theory, Grohe proved the converse to this result: if the problem is solvable in polynomial time, then the class of cores of structures in  $\mathcal{F}$  has bounded treewidth. In particular, this implies that for every  $k > 1$ , there exists some  $k'$  such that if the treewidth of the core of a structure  $\mathbf{A}$  is at least  $k'$ , then the  $k$ -consistency algorithm is not always correct. In his proof, the  $k'$  is an exponential function of  $k$  given by an application of the Excluded Grid Theorem (EGT) of Robertson and Seymour. Instead, our result shows that  $k' = k + 1$  with the additional important feature that our proof does not need the EGT or any conjecture in parameterized complexity theory.

## 2 Preliminaries

*Graphs, structures, and treewidth* A *vocabulary* is a finite set of relation symbols or predicates. Every relation symbol in a vocabulary has an *arity* associated to it. For a vocabulary  $\sigma$ , a *relational structure*  $\mathbf{A}$  of type  $\sigma$  is a pair consisting of a set  $A$ , called the *universe* of  $\mathbf{A}$ , and a sequence of relations  $R^{\mathbf{A}}$ , one for each relation symbol  $R$  from  $\sigma$ , such that the arity of  $R^{\mathbf{A}}$  is equal to that of  $R$ . For example, a graph is a structure with a single binary relation that is symmetric and irreflexive. All structures in this paper are assumed to be *finite*, i.e. having a finite universe.

A structure  $\mathbf{B}$  is called an *induced substructure* of a structure  $\mathbf{A}$  of type  $\sigma$ , if the universe  $B$  of  $\mathbf{B}$  is a subset of the universe  $A$  of  $\mathbf{A}$ , and for any  $R \in \sigma$ ,  $R^{\mathbf{B}} = R^{\mathbf{A}} \cap B^r$ , where  $r$  is the arity of  $R$ .

The *Gaifman graph* of a relational structure  $\mathbf{A} = (A; R_1, \dots, R_n)$  is the graph with vertex set  $A$  and such that  $(a, b)$  is an edge if and only if  $a \neq b$ , and  $a$  and  $b$  belong to the same tuple from one of the relations  $R_1, \dots, R_n$ . Note that loops are never included in the Gaifman graph.

A *tree decomposition* of a graph  $G = (V; E)$  is a labeled tree  $T$  such that

1. every node of  $T$  is labeled by a non-empty subset of  $V$ ,
2. for every edge  $(v, w) \in E$ , there is a node of  $T$  whose label contains  $\{v, w\}$ ,
3. for every  $v \in V$ , the set of nodes of  $T$ , whose labels contain  $v$ , is a subtree of  $T$ .

The *width* of a tree decomposition  $T$  is the maximum cardinality of a label in  $T$  minus 1. The *treewidth* of a graph  $G$  is the smallest number  $k$  such that  $G$  has a tree decomposition of width  $k$ . Note that the treewidth of a tree (containing at least one edge) is one. The treewidth of a structure is the treewidth of its Gaifman graph.

*Homomorphisms, constraint-satisfaction and cores.* A *homomorphism* from a structure  $\mathbf{A}$  to a structure  $\mathbf{B}$  of the same type is a mapping  $f: A \rightarrow B$  between the universes of  $\mathbf{A}$  and  $\mathbf{B}$  such that for every  $r$ -ary  $R \in \sigma$  and every  $(a_1, \dots, a_r) \in R^{\mathbf{A}}$ , we have  $(f(a_1), \dots, f(a_r)) \in R^{\mathbf{B}}$ . The fact that there is a homomorphism from structure  $\mathbf{A}$  to structure  $\mathbf{B}$  we denote by  $\mathbf{A} \rightarrow \mathbf{B}$ . If a homomorphism does not exist we write  $\mathbf{A} \not\rightarrow \mathbf{B}$ .

Let  $\mathbf{A}$  and  $\mathbf{B}$  be two finite relational structures over the same vocabulary  $\sigma$ . We can think of the pair  $\mathbf{A}, \mathbf{B}$  as an instance of the constraint satisfaction problem, where the elements of  $A$  are the variables of the problem, and the elements of  $B$  are the values they may take. A tuple  $(x_1, \dots, x_r) \in R^{\mathbf{A}}$  denotes the constraint that the variables  $x_1, \dots, x_r$  need to take values in  $B$  in such a way that the resulting tuple belongs to  $R^{\mathbf{B}}$ . Therefore, a solution is a mapping  $f: A \rightarrow B$  that defines a homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ .

If  $\mathfrak{A}$  and  $\mathfrak{B}$  are classes of finite relational structures of the same type, the *constraint-satisfaction problem*  $\text{CSP}(\mathfrak{A}, \mathfrak{B})$  asks, given a pair of structures  $\mathbf{A} \in \mathfrak{A}$  and  $\mathbf{B} \in \mathfrak{B}$ , whether or not there is a homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ . If  $\mathfrak{A}$  is the class of all finite structures of a certain type, then we write  $\text{CSP}(*, \mathfrak{B})$  instead of  $\text{CSP}(\mathfrak{A}, \mathfrak{B})$ . Similarly, if  $\mathfrak{B}$  is the class of all finite structures, we write  $\text{CSP}(\mathfrak{A}, *)$ . In addition, if  $\mathfrak{B}$  is one-element, say,  $\mathfrak{B} = \{\mathbf{B}\}$ , then we write  $\text{CSP}(*, \mathbf{B})$ , and similarly for  $\text{CSP}(\mathbf{A}, *)$ .

*Example 1.* Let  $H$  be a (directed) graph. In the  $H$ -COLORING problem we are asked whether there is a homomorphism from a given graph  $G$  to  $H$ . So, the  $H$ -COLORING problem is equivalent to the problem  $\text{CSP}(*, H)$ .

*Example 2.* In the CLIQUE problem we are asked whether a given graph contains a clique of a given size. It is not hard to see that CLIQUE is equivalent to  $\text{CSP}(\mathfrak{K}, *)$ , where  $\mathfrak{K}$  is the class of all finite complete graphs.

An *endomorphism*  $h$  of  $\mathbf{A}$  is a homomorphism from a  $\mathbf{A}$  to itself. Furthermore,  $h$  is said to be an *automorphism* if it is bijective. A structure is a *core* if every endomorphism is an automorphism. A *core* of a relational structure  $\mathbf{A}$  is an induced substructure  $\mathbf{B}$  such that  $\mathbf{A} \rightarrow \mathbf{B}$  and  $\mathbf{B}$  is a core. All cores of a structure are isomorphic, and therefore we can talk about *the core*  $\text{core}(\mathbf{A})$  of a structure  $\mathbf{A}$ . It is easy to see that a structure  $\mathbf{A}$  and its core are *homomorphically equivalent*, meaning that  $\mathbf{A} \rightarrow \text{core}(\mathbf{A})$  and  $\text{core}(\mathbf{A}) \rightarrow \mathbf{A}$ . This allows one to reduce many homomorphism properties of structures and classes of structures, i.e. the complexity of problems  $\text{CSP}(*, \mathfrak{B})$ ,  $\text{CSP}(\mathfrak{A}, *)$ , to the properties of their cores. Yet, with respect to computational complexity, a structure and its core are not always freely exchangeable. In particular, it has been shown that deciding whether a structure is a core is co-NP-complete [15], which implies that, in general, it is hard to compute the core of a structure.

### 3 The $k$ -Consistency Test

Fix some  $k \geq 1$ . The  $k$ -consistency test is a simple algorithm that, given a pair of structures  $\mathbf{A}$  and  $\mathbf{B}$ , either provides a certificate that there is no homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ , or narrows the set of elements of  $\mathbf{B}$  to which each element of  $\mathbf{A}$  may be mapped.

Recall that a solution is a mapping  $f : A \rightarrow B$  that defines a homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ . A partial solution, also called a *partial homomorphism*, is a mapping  $f : A' \rightarrow B$ , where  $A' \subseteq A$ , such that  $f$  defines a homomorphism from the substructure of  $\mathbf{A}$  with universe  $A'$  to the structure  $\mathbf{B}$ . In other words,  $f$  is a function such that for every  $r$ -ary relation symbol  $R \in \sigma$  and  $a_1, \dots, a_r \in A'$ , if  $(a_1, \dots, a_r) \in R^{\mathbf{A}}$  then  $(f(a_1), \dots, f(a_r)) \in R^{\mathbf{B}}$ . If  $f$  and  $g$  are partial solutions we say that  $g$  *extends*  $f$ , denoted by  $f \subseteq g$ , if  $\text{Dom}(f) \subseteq \text{Dom}(g)$  and  $f(a) = g(a)$  for every  $a \in \text{Dom}(f)$ . If  $f \subseteq g$  we also say that  $f$  is the *projection* of  $g$  to  $\text{Dom}(f)$ .

Now we can state the  $k$ -consistency algorithm.

1. Given structures  $\mathbf{A}$  and  $\mathbf{B}$ ;
2. Let  $H$  be the collection of all partial solutions  $f$  with  $|\text{Dom}(f)| \leq k + 1$ ;
3. For every  $f$  in  $H$  with  $|\text{Dom}(f)| \leq k$  and every  $a \in A$ , if there is no  $g$  in  $H$  such that  $f \subseteq g$  and  $a \in \text{Dom}(g)$ , remove  $f$  and all its extensions from  $H$ ;
4. Repeat step 3 until  $H$  is unchanged;
5. If  $H$  is empty reject, else accept.

There are several different but equivalent ways of defining the  $k$ -consistency algorithm. Our formulation is inspired by the *existential  $(k + 1)$ -pebble game* of Kolaitis and Vardi [19]. The connection between the two concepts is due to Kolaitis and Vardi [19].

It is possible to run the algorithm in time polynomial in  $|A|^{k+1}|B|^{k+1}$  because the size of  $H$  in step 2 is bounded by that number, and each iteration removes at least one partial solution. Note that for fixed  $k$ , this is time polynomial in the size of the input. However, if  $k$  is part of the input, the problem of deciding if the  $k$ -consistency test accepts on a given instance is EXP-complete (see [17]).

It is obvious that for any satisfiable instance  $\mathbf{A}, \mathbf{B}$  and any  $k \geq 1$ , the  $k$ -consistency test accepts. The converse is not necessarily true. It holds, for example, if  $k$  is as large as the cardinality of the universe of  $\mathbf{A}$  but it might fail for smaller values of  $k$  (a counterexample easy to verify is given by fixing  $\mathbf{A} = K_{k+2}$  and  $\mathbf{B} = K_{k+1}$ , where  $K_n$  is the clique with  $n$  vertices). The identification of those cases for which the converse is also true is of great interest as it would allow to use the  $k$ -consistency test alone in order to decide the existence of a solution.

**Definition 1.** Let  $\mathfrak{A}$  and  $\mathfrak{B}$  be families of relational structures and let  $k \geq 1$ . We say that  $k$ -consistency solves  $\text{CSP}(\mathfrak{A}, \mathfrak{B})$  if for every  $\mathbf{A} \in \mathfrak{A}$  and  $\mathbf{B} \in \mathfrak{B}$  on which the  $k$ -consistency test accepts, there exists a homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ .

The vast majority of the CSP literature assumes that either  $\mathfrak{A}$  or  $\mathfrak{B}$  is the set of all structures. Although some rather limited results have been obtained in the most general case, a serious attack of this problem seems rather challenging and out of reach by the known techniques.

Observe that  $k$ -consistency solves  $\text{CSP}(*, \mathfrak{B})$  if and only if it solves  $\text{CSP}(*, \mathbf{B})$  for every  $\mathbf{B} \in \mathfrak{B}$ . A similar observation can be made for every  $\mathfrak{A}$ . Consequently, the two main open problems in this research direction can be formulated in the following way:

*Problem 1. ( **$k$ -width problem**)* Characterize all structures  $\mathbf{A}$  for which  $k$ -consistency solves  $\text{CSP}(\mathbf{A}, *)$ . Any such structure is called a  $k$ -width structure. We also say that  $\mathbf{A}$  has  $k$ -width.

*Problem 2. (**width- $k$  problem**)* Characterize all structures  $\mathbf{B}$  for which  $k$ -consistency solves  $\text{CSP}(*, \mathbf{B})$ . Any such structure is called a width- $k$  structure. We also say that  $\mathbf{B}$  has width- $k$ .

For some particular cases, having width- $k$  for some  $k > 1$  is the only reason for polynomial time decidability of a problem. For example, a celebrated result of Hell and Nešetřil [14] asserts that, for a graph  $H$ , if  $H$  is bipartite then  $H$ -COLORING is tractable via the 2-consistency algorithm, and if  $H$  is non-bipartite then  $H$ -COLORING is NP-complete. Later, Nešetřil and Zhu proved, without assuming  $P \neq NP$ , that a finite graph  $H$  has width-2 if and only if  $H$  is bipartite,

A similar statement is not true in the general case of  $\text{CSP}(*, \mathfrak{B})$  [9], and even in the case of  $H$ -COLORING where  $H$  is a digraph [2]: there are constraint-satisfaction problems that are solvable in polynomial time, but not by establishing consistency at any level. An example of this is the problem of checking the consistency of systems of linear equations.

Characterizing those structures having width- $k$ , is a long standing open problem [8], whose solution is only known in a few particular cases of classes of 2- and 3-element structures [21, 4], and of *conservative* structures [5].

## 4 Main Result

This paper addresses and solves completely, in conjunction with [6], the  $k$ -width problem. The following sufficient condition for a structure to have  $k$ -width was identified in [6]:

**Theorem 1** ([6]). *Let  $\mathbf{A}$  be a structure and let  $k \geq 1$ . If  $\text{core}(\mathbf{A})$  has treewidth at most  $k$  then  $\mathbf{A}$  has  $k$ -width.*

Here we prove the exact converse.

**Theorem 2.** *Let  $\mathbf{A}$  be a structure and let  $k \geq 1$ . If  $\mathbf{A}$  has  $k$ -width then  $\text{core}(\mathbf{A})$  has treewidth at most  $k$ .*

Before we prove this, it will be convenient to define the existential pebble game and state the connection with the  $k$ -consistency test first pointed out by Kolaitis and Vardi [19].

The existential  $k$ -pebble game is played between two players, the *Spoiler* and the *Duplicator*, on two relational structures  $\mathbf{A}$  and  $\mathbf{B}$  in accordance with the following rules: on the first round of the game the Spoiler places pebbles on some elements  $a_1, \dots, a_k$  of  $\mathbf{A}$ , and the Duplicator responds by placing pebbles on elements  $b_1, \dots, b_k$  of  $\mathbf{B}$ ; on every further round the Spoiler removes a pebble and places it on another element of  $\mathbf{A}$ , the Duplicator responds by moving the corresponding pebble on  $\mathbf{B}$ . The Spoiler wins if at the end of some round the mapping  $a_i \mapsto b_i, 1 \leq i \leq k$ , is not a partial homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ . The Duplicator wins if he has a *winning strategy*, i.e. a systematic way that allows him to sustain playing “forever”.

Although this presentation of the existential  $k$ -pebble game is certainly very intuitive, it is customary and generally simpler to work with an equivalent “algebraic” definition of the game. The key notion here is that of winning strategy for the Duplicator.

**Definition 2.** *A winning strategy for the Duplicator in the existential  $k$ -pebble game between  $\mathbf{A}$  and  $\mathbf{B}$  is a nonempty collection  $\mathcal{H}$  of partial homomorphisms from  $\mathbf{A}$  to  $\mathbf{B}$  satisfying the following conditions: (a) (restriction condition) if  $f \in \mathcal{H}$  and  $g \subseteq f$ , then  $g \in \mathcal{H}$ ; (b) (extension condition) if  $f \in \mathcal{F}$ ,  $|\text{Dom}(f)| < k$ , and  $a \in A$ , there is  $g \in \mathcal{H}$  such that  $f \subseteq g$  and  $a \in \text{Dom}(g)$ .*

Such a set can be found by starting with the collection of all partial homomorphisms on subsets of at most  $k$  elements, and then removing homomorphisms that do not satisfy one of conditions (a) or (b). Note that this is exactly what the algorithm of the  $(k - 1)$ -consistency test does. Now, it is not difficult to see [19] that the  $k$ -consistency algorithm constructs the most general, i.e., largest, winning strategy for the Duplicator when it exists and reports unsatisfiable when there is no winning strategy. Now we are ready for the proof of the main result:

*Proof of Theorem 2* Let  $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, R_n^{\mathbf{A}})$  be a relational structure, and  $G = G(\mathbf{A})$  its Gaifman graph. Let  $G = (A, E)$ .

Since  $\mathbf{A}$  has  $k$ -width if and only if  $\text{core}(\mathbf{A})$  has  $k$ -width, we may assume that  $\mathbf{A} = \text{core}(\mathbf{A})$ .

$E$  is a symmetric and irreflexive binary relation on  $A$ . We denote edges of  $G$  by unordered pairs  $e = \{a, a'\}$ . Let  $a_0 \in A$  be a distinguished point of  $A$  to be defined later. For every  $a \in A$ , let  $d_a$  denote the degree of  $a$  in  $G$ , and let  $e_1^a, \dots, e_{d_a}^a$  be a fixed enumeration of all the edges that are incident on  $a$ .

Let  $\mathbf{B} = \mathbf{B}(\mathbf{A})$  be the relational structure defined as follows. The set of vertices of  $\mathbf{B}$  is the set of all tuples of the form  $(a, (b_1, \dots, b_{d_a}))$ , where

1.  $a \in A$  and  $b_1, \dots, b_{d_a} \in \{0, 1\}$ ,
2.  $b_1 + \dots + b_{d_a} \equiv 0 \pmod{2}$  if  $a \neq a_0$ ,
3.  $b_1 + \dots + b_{d_a} \equiv 1 \pmod{2}$  if  $a = a_0$ .

A tuple  $((a^1, (b_1^1, \dots, b_{d_{a^1}}^1)), \dots, (a^n, (b_1^n, \dots, b_{d_{a^n}}^n)))$  belongs to the  $(n$ -ary) relation  $R_i^{\mathbf{B}}$  if and only if

1. the tuple  $(a^1, \dots, a^n)$  belongs to  $R_i^{\mathbf{A}}$ ,
2. if  $\ell, m, i$  and  $j$  are such that  $\{a^\ell, a^m\} = e_i^{a^\ell} = e_j^{a^m}$ , then  $b_i^\ell = b_j^m$ .

Intuitively, each vertex of  $\mathbf{B}$  is an assignment of 0/1-values to the edges of  $G(\mathbf{A})$  incident to a given point  $a \in A$ , and the tuples from relations of  $\mathbf{B}$  encode the constraints that the assignments of values to the edges of two adjacent points  $a, a' \in A$  must be consistent.

*Example 3.* Let  $\mathbf{A}$  be a clique with vertices  $a, b$ , and  $c$ . If we choose the distinguish vertex  $a_0$  to be  $a$ , the structure  $\mathbf{B}(\mathbf{A})$  is the graph with the vertex set  $\{(a, (0, 1)), (a, (1, 0)), (b, (0, 0)), (b, (1, 1)), (c, (0, 0)), (c, (1, 1))\}$  shown in the picture.

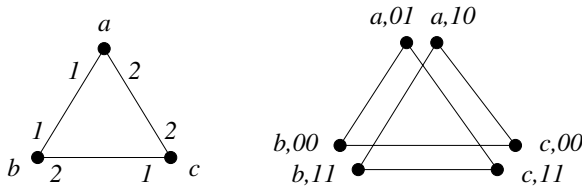


Fig. 1.  $\mathbf{A}$  and  $\mathbf{B}(\mathbf{A})$

Note that the first projection  $\pi : B \rightarrow A$ , defined by  $\pi((a, (b_1, \dots, b_{d_a}))) = a$  is a homomorphism from  $\mathbf{B}$  to  $\mathbf{A}$ .

**Lemma 1.** *If  $\mathbf{A}$  is a core, then there is no homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ .*

*Proof.* Suppose that  $\mathbf{A}$  is a core, and suppose for contradiction that  $h : A \rightarrow B$  is a homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ . Let  $\pi : B \rightarrow A$  be the first projection. Composing,  $f = h \circ \pi$  is a homomorphism from  $\mathbf{A}$  to  $\mathbf{A}$ , and since  $\mathbf{A}$  is a core, it must be an automorphism. Now, let  $g = h \circ f^{-1}$  and note that  $g$  is

still a homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$  with the additional property that  $g(a) = (a, (b_1^a, \dots, b_{d_a}^a))$  for some  $b_1^a, \dots, b_{d_a}^a \in \{0, 1\}$  and every  $a \in A$ .

Now, for every edge  $e = \{a, a'\}$  of  $G(\mathbf{A})$ , define  $x_e = b_i^a = b_j^{a'}$ , where  $i$  and  $j$  are such that  $e_i^a = e$  and  $e_j^{a'} = e$ . The equality  $b_i^a = b_j^{a'}$  follows from the fact that  $g$  is a homomorphism. Now, we have

$$x_{e_1^a} + \dots + x_{e_{d_a}^a} \equiv 0 \pmod{2}$$

for every  $a \neq a_0$ , and

$$x_{e_1^a} + \dots + x_{e_{d_a}^a} \equiv 1 \pmod{2}$$

for  $a = a_0$ . Since every edge of  $G(\mathbf{A})$  has exactly two end-points, adding up all equations we get

$$2 \sum_e x_e \equiv 1 \pmod{2};$$

a contradiction.

We need an alternative definition of treewidth. Let  $G = (V, E)$  be a graph. We say that two sets  $B, C \subseteq V$  *touch* if either they intersect or there is an edge of  $G$  with an end-point in  $B$  and the other in  $C$ . A *bramble* is a collection  $B_1, \dots, B_r$  of pairwise touching connected subsets of  $G$ . A *cover* of this bramble is a set of points that intersects every  $B_i$ . The *order* of a bramble is the minimum size of its covers. Seymour and Thomas [22] proved that a connected graph has treewidth at least  $k$  if and only if it has a bramble of order at least  $k + 1$ .

**Lemma 2.** *If the treewidth of  $\mathbf{A}$  is at least  $k + 1$ , then the Duplicator wins the existential  $k + 1$ -pebble game on  $\mathbf{A}$  and  $\mathbf{B}$ .*

*Proof.* We start with some definitions. For every walk  $P = (a_0, a_1, \dots, a_r)$  in  $G(\mathbf{A})$  that starts at  $a_0$  and for every edge  $e$  of  $G(\mathbf{A})$ , we define

1.  $x_e^P = 1$  if  $e$  appears an odd number of times in  $P$ ,
2.  $x_e^P = 0$  if  $e$  appears an even number of times in  $P$ .

Now we define  $h^P(a) = (a, (x_{e_1^a}^P, \dots, x_{e_{d_a}^a}^P))$  for every  $a \in A$ .

*Claim.* If  $P = (a_0, a_1, \dots, a_r)$  is a walk in  $G(\mathbf{A})$  that starts at  $a_0$  and  $a \neq a_r$ , then  $h^P(a)$  belongs to  $B$ .

*Proof.* Suppose that  $P = (a_0, a_1, \dots, a_r)$  is a walk in  $G(\mathbf{A})$  that starts at  $a_0$ , and let  $a \neq a_r$ . We need to check that

$$x_{e_1^a}^P + \dots + x_{e_{d_a}^a}^P \equiv 0 \pmod{2} \tag{1}$$

if  $a \neq a_0$ , and

$$x_{e_1^a}^P + \dots + x_{e_{d_a}^a}^P \equiv 1 \pmod{2} \tag{2}$$



if  $a = a_0$ . Suppose first that  $a \neq a_0$ . Let  $i_1 < \dots < i_s$  be the enumeration of all positions  $i$  in the walk  $P = (a_0, a_1, \dots, a_r)$  with  $a_i = a$ . Since the walk does not start or end at  $a$ , we have  $0 < i_1 < \dots < i_s < r$ . It follows immediately that the total number of occurrences of edges in the walk that are incident on  $a$  is even (we are using the fact that  $G(\mathbf{A})$  has no self-loops so for every  $1 \leq j \leq r - 1$  there exists  $l$  such that  $i_j < l < i_{j+1}$ ). Thus, equation (II) holds. Suppose now that  $a = a_0$ . Again, let  $i_1 < \dots < i_s$  be the enumeration of all positions  $i$  in the walk such that  $a_i = a$ . Since the walk starts at  $a_0 = a$  and does not end at  $a$ , we have  $0 = i_1 < \dots < i_s < r$ . It follows immediately that the total number of occurrences of edges in the walk that are incident on  $a$  is odd. Thus, equation (I) holds.

*Claim.* Let  $S \subseteq A$ , and let  $P = (a_0, a_1, \dots, a_r)$  be a walk in  $G(\mathbf{A})$  that starts at  $a_0$  and does not end in a point in  $S$ . Then, the restriction  $h^P|_S$  of  $h^P$  to  $S$  is a partial homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ .

*Proof.* The previous claim guarantees that  $h^P(a)$  belongs to  $B$  for every  $a \in S$ . We need to check now that for every ( $n$ -ary) relation  $R_i^{\mathbf{A}}$  and any tuple  $(b_1, \dots, b_n) \in R_i^{\mathbf{A}}$  such that  $b_1, \dots, b_n \in S$ , the tuple  $(h^P(b_1), \dots, h^P(b_n))$  is also a tuple from  $R_i^{\mathbf{B}}$ . But this is obvious because for any  $j$  and  $\ell$  if  $s$  and  $t$  are such that  $e_s^{b_j} = e_t^{b_\ell} = \{b_j, b_\ell\}$ , then trivially  $x_{e_s^{b_j}}^P = x_{e_t^{b_\ell}}^P = x_{\{b_j, b_\ell\}}^P$ .

Now we can define the winning strategy for the Duplicator in the existential  $k + 1$ -pebble game between  $\mathbf{A}$  and  $\mathbf{B}$ . Suppose that the treewidth of  $\mathbf{A}$  is at least  $k + 1$ . Let  $\{B_1, \dots, B_r\}$  be a bramble of  $\mathbf{A}$  of order at least  $k + 2$ . It is finally the time to define  $a_0$ . Let us fix  $a_0$  to be any point of  $A$  connected to the bramble. We define a collection of partial homomorphisms  $\mathcal{H}$  as follows: for any walk  $P$  in  $G(\mathbf{A})$  that starts with  $a_0$  and any  $S \subseteq A$  such that  $|S| \leq k + 1$  and the last vertex of  $P$  belongs to a bag of the bramble that does not contain any element of  $S$ , we include  $h^P|_S$  into  $\mathcal{H}$ . By the claims above, each such  $h^P|_S$  is indeed a partial homomorphism. The election of  $a_0$  guarantees that  $\mathcal{H}$  is nonempty. Conditions (a),(b) from the definition of a winning strategy can be easily checked. Condition (a) holds trivially. For (b), let  $h_S^P$  be any function with  $|S| \leq k$  in  $\mathcal{H}$ . Hence  $P$  is a walk that starts at  $a_0$  and ends at a point, say  $a_r$ , that sits in a bag  $B_i$  that does not contain any point in  $S$ . Now let  $a \in A$ . Let  $B_j$  be a bag of the bramble that does not contain any point in  $S' = S \cup \{a\}$ . Such a bag also exists because  $|S'| \leq k + 1$  and the bramble cannot be covered with less than  $k + 2$  points. Since all pairs of bags of the bramble touch, and since bags are connected, there must be a walk  $Q$  from  $a_r$  to a point in  $B_j$  that runs entirely inside  $B_i$  except for the last point, which lands in  $B_j$ . Now we let  $P'$  be the concatenation of  $P$  with  $Q$ . The walk  $P'$  has the properties we need: it starts at  $a_0$  and it ends in a point that belongs to a bag  $B_j$  of the bramble that does not contain any point in  $S'$ . Thus  $h^{P'}|_{S'}$  belongs to  $\mathcal{H}$ . Finally, since the only edges that  $P'$  adds to  $P$  are edges that are entirely inside  $B_i$  except for the last that may land inside  $B_j$ , none of these edges has an end-point in  $S$  because both  $B_i$  and  $B_j$  are  $S$ -free. It follows that  $x_e^P = x_e^{P'}$  for every edge  $e$  with an end-point in  $S$ , so  $h^{P'}(a) = h^P(a)$  for every  $a \in S$ .

## 5 Further Comments and Remarks

If we put Theorem 1 and Theorem 2 together we obtain that  $\mathbf{A}$  has  $k$ -width if and only if  $\text{core}(\mathbf{A})$  has treewidth at most  $k$ . In turn, it was proved in [6] that for every fixed  $k \geq 1$ , it is an NP-complete problem to decide if a given structure has a core of treewidth at most  $k$ . This implies that it is an NP-complete problem to decide if a given structure has  $k$ -width. Before our result, it was not even known whether this problem was decidable. Dually, it is an important open problem whether it is decidable if a given structure has width- $k$ .

The second remark is about an application of our result to preservation theorems in finite model theory. Let us first note that, for a core  $\mathbf{A}$  of treewidth at least  $k$ , the proof of our main result provides a structure  $\mathbf{B}$  with the following three properties:  $\mathbf{B} \rightarrow \mathbf{A}$ ,  $\mathbf{A} \not\rightarrow \mathbf{B}$ , and the Duplicator wins the existential  $k$ -pebble game on  $\mathbf{A}$  and  $\mathbf{B}$ . Using these three properties, it is possible to solve an open problem in [3]. The problem asked whether every sentence of first-order logic that can be written equivalently as an existential-positive infinitary sentence with  $k$  variables on finite structures is also equivalent to a existential-positive finite sentence with  $k$  variables on finite structures. The construction above, in combination with Rossman's Theorem [20], provides the right tool to establish this preservation theorem. We will provide the details of the proof in the journal version of this paper.

It would be interesting to see if the construction we give has more applications in finite model theory. Interestingly, up to now we have used it to establish both a *negative* result (limits on the  $k$ -consistency algorithm), and a *positive* result (a preservation theorem in finite model theory).

## Acknowledgements

Part of this work was done when all authors visited the Isaac Newton Institute for Mathematical Sciences, Cambridge under the Logic and Algorithms program. The first author is supported in part by CICYT TIN-2004-04343. The second author is supported by a NSERC Discovery grant. The third author is supported by the MEC under the program "Ramon y Cajal", grants TIN 2005-09312-C03-03, TIN 2004-04343, the EU PASCAL Network of Excellence IST-2002-506778, and the MODNET Marie Curie Research Training Network MRTN-CT-2004-512234.

## References

1. Atserias, A., Dalmau, V.: A Combinatorial Characterization of ResolutionWidth. In: Proceedings of the 18th Annual IEEE Conference on Computational Complexity, Aarhus, Denmark, July 2003, pp. 239–247 (2003)
2. Atserias, A.: On digraph coloring problems and treewidth duality. In: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science, Chicago, June 2005, pp. 106–115 (2005)
3. Atserias, A., Dawar, A., Kolaitis, P.: On Preservation under Homomorphisms and Unions of Conjunctive Queries. *Journal of the ACM* 53(2), 208–237 (2006)

4. Bulatov, A.: A dichotomy theorem for constraints on a three-element set. *J. of the ACM* 53(1), 66–120 (2006)
5. `spaceskip=.28em plus .1em minus .1em`Bulatov, A.A.: Tractable conservative constraint satisfaction problems. In: *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science*, Ottawa, Canada, June 2003, pp. 321–330. IEEE Computer Society Press, Los Alamitos (2003)
6. Dalmau, V., Kolaitis, P.G., Vardi, M.Y.: Constraint satisfaction, bounded treewidth, and finite variable logics. In: Van Hentenryck, P. (ed.) *CP 2002*. LNCS, vol. 2470, pp. 311–326. Springer, Heidelberg (2002)
7. Dechter, R.: From local to global consistency. *Artificial Intelligence* 55(1), 87–107 (1992)
8. Feder, T., Vardi, M.Y.: Monotone monadic SNP and constraint satisfaction. In: *Proceedings of 25th ACM Symposium on the Theory of Computing (STOC)*, pp. 612–622. ACM Press, New York (1993)
9. Feder, T., Vardi, M.Y.: The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal of Computing* 28, 57–104 (1998)
10. Freuder, E.: A Sufficient Condition for Backtrack-Free Search. *Journal of the ACM* 29(1), 24–32 (1982)
11. Freuder, E.: Complexity of  $k$ -tree structured constraint satisfaction problems. In: *Proceedings of the 8th National Conference on Artificial Intelligence AAAI-90*, pp. 4–9 (1990)
12. Grohe, M.: The complexity of homomorphism and constraint satisfaction problems seen from the other side. In: *Proceedings of the 44th Annual Symposium on Foundations of Computer Science*, Cambridge, Massachusetts, October 2003, pp. 552–561. IEEE Computer Society Press, Los Alamitos (2003)
13. Hell, P., Nešetřil, J.: *Graphs and homomorphisms*. Oxford Lecture Series in Mathematics and its Applications, vol. 28. Oxford University Press, Oxford (2004)
14. Hell, P., Nešetřil, J.: On the complexity of  $H$ -coloring. *Journal of Combinatorial Theory, Ser.B* 48, 92–110 (1990)
15. Hell, P., Nešetřil, J.: The core of a graph. *Discrete Mathematics* 109(1-3), 117–126 (1992)
16. Hell, P., Nešetřil, J., Zhu, X.: Duality and polynomial testing of tree homomorphisms. *Trans. of the AMS* 348(4), 1281–1297 (1996)
17. Kolaitis, Ph.G., Panttaja, J.: On the Complexity of Existential Pebble Games. In: *Proceeding of the 17th International Workshop on Computer Science Logic*, pp. 314–329 (2003)
18. Kolaitis, Ph.G., Vardi, M.Y.: On the expressive power of Datalog: tools and case study. *Journal of Computer and System Sciences* 51(1), 110–134 (1995)
19. Kolaitis, P.G., Vardi, M.Y.: A game-theoretic approach to constraint satisfaction. In: *Proceedings of the 17th National (US) Conference on Artificial Intelligence, AAAI'00*, pp. 175–181 (2000)
20. Rossman, B.: Existential Positive Types and Preservation under Homomorphisms. In: *Proceedings of the 20th IEEE Symposium on Logic in Computer Science*, pp. 467–476. IEEE Computer Society Press, Los Alamitos (2005)
21. Schaefer, T.J.: The complexity of satisfiability problems. In: *Proceedings of the 10th ACM Symposium on Theory of Computing (STOC'78)*, pp. 216–226. ACM Press, New York (1978)
22. Seymour, P., Thomas, R.: Graph searching, and a min-max theorem for treewidth. *Journal of Combinatorial Theory, Series B* 58, 22–23 (1993)

# Complexity of Propositional Proofs Under a Promise<sup>\*</sup>

Nachum Dershowitz<sup>1,2</sup> and Iddo Tzameret<sup>1</sup>

<sup>1</sup> School of Computer Science, Tel Aviv University, Ramat Aviv 69978, Israel

<sup>2</sup> Microsoft Research, Redmond, WA 98052, USA

{nachumd,tzameret}@post.tau.ac.il

*“Goods Satisfactory or Money Refunded”* – The Eaton Promise

**Abstract.** We study – within the framework of propositional proof complexity – the problem of certifying unsatisfiability of CNF formulas under the promise that any satisfiable formula has many satisfying assignments, where “many” stands for an explicitly specified function  $\Lambda$  in the number of variables  $n$ . To this end, we develop propositional proof systems under different measures of promises (that is, different  $\Lambda$ ) as extensions of resolution. This is done by augmenting resolution with axioms that, roughly, can eliminate sets of truth assignments defined by Boolean circuits. We then investigate the complexity of such systems, obtaining an exponential separation in the average-case between resolution under different size promises:

- (i) Resolution has polynomial-size refutations for all unsatisfiable 3CNF formulas when the promise is  $\varepsilon \cdot 2^n$ , for any constant  $0 < \varepsilon < 1$ .
- (ii) There are no sub-exponential size resolution refutations for random 3CNF formulas, when the promise is  $2^{\delta n}$  (and the number of clauses is  $o(n^{3/2})$ ), for any constant  $0 < \delta < 1$ .

**Keywords:** proof complexity, resolution, random 3CNF, promise problems.

## 1 Introduction

Demonstrating unsatisfiability of propositional formulas is one of the most fundamental problems in complexity theory, as well as in hardware and software validation. Any standard sound and complete propositional proof system has the ability to separate the set of unsatisfiable formulas in conjunctive normal form (CNF) from the set of CNF formulas having at least one satisfying assignment, in the sense that every unsatisfiable CNF has a refutation in the system, while no satisfiable CNF has one. Our goal is to develop and study, within the framework of propositional proof complexity, systems that are “sound and complete” in a relaxed sense: they can separate the set of unsatisfiable CNF formulas

---

\* This work was carried out in partial fulfillment of the requirements for the Ph.D. degree of the second author and was supported in part by the Israel Science Foundation (grant no. 250/05).

from the set of CNF formulas having *sufficiently many* satisfying assignments (where the term “sufficiently many” stands for an explicitly given function of the number of variables in the CNF). We call such proof systems *promise refutation systems*, as they are complete and sound for the set of CNF formulas promised to be either unsatisfiable or to have many satisfying assignments.

As the proof systems we develop here intend to prove the *unsatisfiability* of CNF formulas (in other words, to *refute* them, which is to validate their negation), we will work solely with *refutation* systems, and shall speak about refutations and proofs interchangeably, always intending refutations, unless otherwise stated. In particular, we work with refutation systems that extend the widely studied resolution refutation system.

Our first task is to introduce a natural model for promise propositional refutation systems. This is accomplished by augmenting the standard resolution refutation system (or any other propositional proof system extending resolution) with an additional collection of axioms, the *promise axioms*. Each refutation in a promise refutation system can make use of at most one promise axiom. The promise axioms are meant to capture the idea that we can “ignore” or “discard” a certain number of truth assignments from the space of all truth assignments, and still be able to certify (due to the promise) whether the given CNF is unsatisfiable or not. The number of assignments that a promise axiom is allowed to discard depends on the promise we are given, and specifically it needs to be less than the number of assignments promised to satisfy a given CNF (unless it is unsatisfiable).

Assuming we have a promise that a satisfiable CNF has more than  $\Lambda$  satisfying assignments then we can discard up to  $\Lambda$  assignments. We refer to  $\Lambda$  as the *promise*. This way the refutation system is guaranteed not to contain refutations of CNF formulas having more than  $\Lambda$  satisfying assignments, as even after discarding (at most  $\Lambda$ ) assignments we still have at least one satisfying assignment left, and on the other hand, any unsatisfiable CNF formula has a refutation in the system, as resolution already has a refutation of it. We now explain (somewhat informally) what it means to *discard* assignments and how promise axioms formulate the notion of discarding the *correct number* of truth assignments.

Essentially, we say that a truth assignment  $\mathbf{a}$  is discarded by some Boolean formula if  $\mathbf{a}$  falsifies the formula. More formally, let  $X := \{x_1, \dots, x_n\}$  be the set of the underlying variables of a given CNF, called the *original variables*. Let  $A$  be some CNF formula in the  $X$  variables, and assume that  $A$  also contains variables not from  $X$  called *extension variables*. Let  $\mathbf{a} \in \{0, 1\}^n$  be a truth assignment to the  $X$  variables, and assume that there is no extension of  $\mathbf{a}$  (to the extension variables) that satisfies  $A$ . Thus, any assignment satisfying  $A$  must satisfy also  $X \not\equiv \mathbf{a}$  (that is,  $A \models X \not\equiv \mathbf{a}$ ), and so any (implicational) complete proof system can prove  $X \not\equiv \mathbf{a}$  from  $A$ , or, in the case of a refutation system, can refute  $X \equiv \mathbf{a}$ , given  $A$ . In this case, we say that the assignment  $\mathbf{a}$  is *discarded by  $A$* .

The promise axioms we present have two main properties:

- (I) They discard assignments from the space of possible assignments to the variables  $X$ .
- (II) They express the fact that not too many assignments to the variables  $X$  are being discarded (in a manner made precise).

The first property is achieved as follows: Let  $C$  be any Boolean circuit with  $n$  output bits. Then we can formulate a CNF formula denoted by  $A$  (using extension variables) expressing the statement that the (vector of) variables  $X$  is equal to the output of  $C$ . This enables  $A$  to *discard every truth assignment to the  $X$  variables outside the image of the Boolean map defined by  $C$*  (as if an assignment  $\mathbf{a}$  to the  $X$  variables is not in the image of  $C$  then no extension of  $\mathbf{a}$  can satisfy  $A$ , assuming the formulation of  $A$  is correct). (The actual definition is a bit different than described here, due to technical reasons; see Sect. 3).

The second property is achieved as follows: Assume we can make the statement that the *domain* of the map defined by the Boolean circuit  $C$  above is of size at least  $2^n - \Lambda$  explicit (see Sect. 3 for more details on this). Then, in order for the second property to hold it is sufficient that the axiom formulates the statement that the circuit  $C$  defines an *injective* map (and thus the image of the map contains enough truth assignments), which can be done quite naturally.

Given a certain promise and its associated promise axiom, we call a refutation of resolution augmented with the promise axiom a *resolution refutation under the (given) promise*.

Our second task, besides introducing the model of promise refutation systems, is to investigate the basic properties of this model and in particular to determine its average-case proof complexity with respect to different size of promises (see below for a summary of our findings in this respect).

## 1.1 Background and Motivation

In propositional proof complexity theory, it is standard to consider an *abstract* or *formal* propositional proof system (usually called a *Cook-Reckhow proof system*, following [3]) as a polynomial-time algorithm  $A$  that receives a Boolean formula  $F$  (usually in CNF) and a string  $\pi$  over some finite alphabet (“the (proposed) refutation” of  $F$ ), such that there exists a  $\pi$  with  $A(F, \pi) = 1$  if and only if  $F$  is unsatisfiable. (A string  $\pi$  for which  $A(F, \pi) = 1$  is also called a *witness* for the unsatisfiability of  $F$ .) Equipped with this abstract definition of propositional proof systems, showing that *for every* abstract proof system there exists some family of formulas  $F$  for which there is no polynomially-bounded family of proofs  $\pi$  of  $F$  is equivalent to showing  $\mathbf{NP} \neq \mathbf{coNP}$ .

For this reason (among others), it is customary in proof complexity theory to concentrate on specific (sometimes provably weaker) proof systems for which proofs have a simple structure. This makes the complexity analysis of such proof systems simpler. Prominent examples of such systems are Frege systems and weaker subsystems of Frege, the most notable of which is the resolution refutation system, which also plays an important rôle in many automated theorem provers.

In accordance with this, we shall be interested not with abstract promise proof systems (that is, not with finding general witnesses for unsatisfiability, possibly under a promise), but rather with specific and more structured proof systems, and specifically with refutation systems built-up as extensions of resolution.

A natural relaxation of the problem of unsatisfiability certification is to require that, if a CNF is satisfiable, then it actually have many satisfying assignments. As mentioned above, we call the specific number of assignments (as a function of the number of variables  $n$ ) required to satisfy a satisfiable CNF formula, the promise. Accordingly, one can define an *abstract promise proof system* in an analogous manner to the definition of an abstract proof system. It is thus natural to ask whether giving such a promise can help in obtaining shorter proofs of unsatisfiability.

In the case of a *big* promise, that is, a constant fraction of the space of all truth assignments ( $\Lambda = \varepsilon \cdot 2^n$ , for a constant  $0 < \varepsilon < 1$ ), there is already a *deterministic polynomial-time algorithm* for any fixed natural number  $k$  that certifies the unsatisfiability of all unsatisfiable  $k$ CNF formulas under the promise: The algorithm receives a  $k$ CNF that is either unsatisfiable or has more than  $\Lambda$  satisfying assignments and answers whether the formula is unsatisfiable (in case the formula is satisfiable, the algorithm provides a satisfying assignment); see the papers by Hirsch [6] and Trevisan [7] for such efficient algorithms. This trivially implies the existence of polynomial-size witnesses for any unsatisfiable  $k$ CNF under the promise  $\varepsilon \cdot 2^n$ . But does already resolution admit such short witnesses of unsatisfiability (that is, resolution refutations) under a big promise? We show that the answer is positive (for all unsatisfiable 3CNF formulas).

In the case of a *smaller* promise, by which we mean  $\Lambda = 2^{\delta n}$  for a constant  $0 < \delta < 1$ , it is possible to efficiently transform any CNF over  $n$  variables to a new CNF with  $n' = \lceil n/(1 - \delta) \rceil$  variables such that the original CNF is satisfiable if and only if the new CNF has at least  $2^{\delta n'}$  satisfying assignments. This can be achieved by adding “dummy variables” (e.g. variables that do not occur at all in the formula or by adding any satisfiable CNF consisting of these dummy variables to the original CNF). Thus, the *worst-case* complexity of certifying CNF unsatisfiability under such a promise is polynomially equivalent to the worst-case complexity of certifying CNF unsatisfiability without a promise. However, it is still possible that a promise of  $2^{\delta n}$  might give some advantage (that is, a super-polynomial speedup over refutations without a promise) in certifying the unsatisfiability of certain (but not all) CNF formulas; for instance, in the *average-case*.<sup>1</sup>

Feige, Kim and Ofek [5] showed that when the number of clauses is  $\Omega(n^{7/5})$  there exist polynomial-size witnesses that witness the unsatisfiability of 3CNF

<sup>1</sup> Note that if we add dummy variables to a 3CNF then we obtain an “atypical instance” of a 3CNF. Thus, assuming we have polynomial-size witnesses of unsatisfiability of 3CNF formulas under a small promise in the average-case (the “typical case”), the reduction alone (that is, adding dummy variables) does *not* automatically yield polynomial-size witnesses for 3CNF formulas in the average-case without a promise as well.

formulas in the *average-case*. On the other hand, Beame et al. [1] and Ben-Sasson and Wigderson [2] showed that resolution does not provide sub-exponential refutations for 3CNF formulas in the average-case when the number of clauses is at most  $n^{(3/2)-\epsilon}$ , for any constant  $0 < \epsilon < 1/2$ . This shows that general witnessing of 3CNF unsatisfiability is strictly stronger than resolution refutations. But is it possible that under a promise of  $2^{\delta n}$  resolution can do better in the average-case? We show that the answer is negative.

There are two main motivations for studying propositional proofs under a given promise and their complexity. The first is to answer the natural question whether CNF unsatisfiability certification enjoys any advantage given a certain promise. As already mentioned, the answer is positive when the promise is a constant fraction of all the truth assignments, and our results imply that this phenomenon already occurs for resolution. For a smaller promise of  $2^{\delta n}$ , we can show that at least in the case of resolution refutations of most 3CNF formulas (of certain clause-to-variable density) the answer is negative. In fact, we can show that the answer stays negative even when the promise is bigger than  $2^{\delta n}$ , and specifically when  $\Lambda = 2^n/2^{n^\xi}$  for some constant  $0 < \xi < 1$ . Overall, our results establish the first unsatisfiability certification model in which a promise of a certain size is known to help (i.e., allows for more efficient certifications) in the average-case, while a promise of smaller size does not help.

The second motivation, is more intrinsic to proof complexity theory. It is a general goal to develop natural frameworks for propositional proofs that are not sound in the strict sense, but rather possess an approximate notion of soundness (like showing that certain “approximations” give speed-ups). For this purpose, the proof systems we propose formalize – in a natural way – the notion of separating unsatisfiable CNF formulas from those that have many satisfying assignments. The promise axioms we present also allow for a natural way of controlling the size of the promise, which in addition leads to an exponential separation between different size promises.

This paper introduces the concept of propositional proofs under a promise, analyzes the proof complexity of these proof systems with respect to different promise sizes, giving a separation between promises of different sizes, and also illustrates several new facts about the widely studied resolution proof system.

## 1.2 Results

We show that resolution refutations are already enough to efficiently separate unsatisfiable 3CNF formulas from those 3CNF formulas with an arbitrarily small constant fraction of satisfying assignments. In particular, in Section 4, we show the following:

**First Main Result:** Let  $0 < \epsilon < 1$  be some constant and let  $\Lambda = \epsilon \cdot 2^n$  be the given promise. Then *every* unsatisfiable 3CNF with  $n$  variables has a polynomial-size (in  $n$ ) resolution refutation under the promise  $\Lambda$ .

The proof resembles a deterministic algorithm of Trevisan [7] for approximating the number of satisfying assignments of  $k$ CNF formulas.



In contrast to the case of a big promise, we also show that, at least for resolution, a small promise of  $\Lambda = 2^{\delta n}$  (for any constant  $0 < \delta < 1$ ) does not give any advantage over standard resolution (resolution without the promise axioms) in most cases (that is, in the average-case). Specifically, in Section 5, we show the following:

**Second Main Result:** Let  $0 < \delta < 1$  be any constant and let  $\Lambda = 2^{\delta n}$  be the given promise. Then, there is an exponential lower bound on the size of resolution refutations of random 3CNF formulas under the promise  $\Lambda$ , when the number of clauses is  $o(n^{3/2})$ .

This lower bound actually applies to a more general model of promise proofs: It remains valid even if we allow (somehow) the promise proofs to discard *arbitrarily chosen* sets of truth assignments (of  $\Lambda = 2^{\delta n}$  size), and not necessarily those sets that are definable by (small) Boolean circuits. In fact, the lower bound applies even to a bigger promise of  $\Lambda = 2^{n-n^\xi}$ , for some constant  $0 < \xi < 1$ .

The proof strategy of this lower bound follows that of Ben-Sasson and Wigderson [2] (the *size-width tradeoff* approach), and so the rate of the lower bound matches the one in that paper. The main novel observation is that under the appropriate modifications this strategy also works when one restricts the set of all truth assignments to a smaller set (that is, from  $2^n$  down to  $2^n - 2^{\delta n}$  for a constant  $0 < \delta < 1$ , and in fact down to  $2^n - 2^{n-n^\xi}$ , for some constant  $0 < \xi < 1$ ).

It is important to note that the two main results above show that the decision to discard sets of truth assignments defined by *Boolean circuits* does not effect the results in any way, and thus should not be regarded as a restriction of the model of promise refutations (at least not for resolution). To see this, note that we could allow a promise refutation to discard *arbitrarily chosen* sets of truth assignments (of the appropriate size determined by the given promise); that is, sets of truth assignments that are not necessarily definable by (small) Boolean circuits. However, although this modification strengthens the model it is not really necessary for the *upper bound* in the First Main Result, as this upper bound is already valid when one discards sets of truth assignments by (small) Boolean circuits. On the other hand, as mentioned above, the *lower bound* in the Second Main Result is already valid when one allows a promise refutation to discard any *arbitrarily chosen* set of truth assignments (of the appropriate size).

The exact model of promise propositional proof systems is developed in Sect. 3. It is preceded by preliminaries and terminological conventions.

## 2 Preliminaries

**Resolution refutation system.** Resolution is a complete and sound proof system for unsatisfiable CNF formulas. Let  $C$  and  $D$  be two clauses containing neither  $x_i$  nor  $\neg x_i$ , the *resolution rule* allows one to derive  $C \vee D$  from  $C \vee x_i$  and  $D \vee \neg x_i$ . The clause  $C \vee D$  is the *resolvent* of the clauses  $C \vee x_i$  and  $D \vee \neg x_i$  on the variable  $x_i$ . The *weakening rule* allows to derive the clause  $C \vee D$  from the clause  $C$ , for any two clauses  $C, D$ .

**Definition 1 (Resolution).** A resolution proof of the clause  $D$  from a CNF formula  $K$  is a sequence of clauses  $D_1, D_2, \dots, D_\ell$ , such that: (1) each clause  $D_j$  is either a clause of  $K$  or a resolvent of two previous clauses in the sequence or derived by the weakening rule from a previous clause in the sequence; (2) the last clause  $D_\ell = D$ . The size of a resolution proof is the total number of clauses in it. A resolution refutation of a CNF formula  $K$  is a resolution proof of the empty clause  $\square$  from  $K$  (the empty clause stands for FALSE).

Let  $K$  be an unsatisfiable CNF formula. The *resolution refutation size* of  $K$  is the minimal size of a resolution refutation of  $K$ . If  $K$  has a polynomial-size resolution refutation we say that resolution can *efficiently certify* the unsatisfiability of  $K$ . Similarly, if the clause  $D$  has a polynomial-size resolution proof from  $K$  we say that  $D$  is *efficiently provable* from  $K$ .

**Boolean circuit encoding.** The promise axioms we introduce use Boolean circuits to define the set of assignments to be discarded (see Sect. 3). Therefore, as resolution operates only with clauses, we need to encode Boolean circuits as collections of clauses (CNF formulas). For most purposes, we will not need an explicit description of how this encoding is done. Nevertheless, in Sect. 4 we need to ensure that resolution can efficiently prove several basic facts about the encoded circuits. For this reason, and for the sake of concreteness of the promise axioms (see Definition 4), we provide the precise definition of the encoding in the full version of this paper [4], in addition to proving some of the encoding's basic (proof theoretical) properties.

### 3 Promise Proof Systems

In this section, we define precisely the model of refutations under a promise. As discussed in the introduction, we work with the resolution refutation system as our underlying system and augment it with a new set of axioms that we call the *promise axioms*. We call this proof system *promise resolution*. The promise axioms are meant to express the fact that we can discard a certain number of truth assignments from the space of all truth assignments and still be able to certify (due to the promise) whether the input CNF is unsatisfiable or not. Each promise resolution refutation can use at most one promise axiom.

From now on, we assume that the underlying variables of the CNF formulas that are meant to be refuted are taken from the set  $X := \{x_1, \dots, x_n\}$ . The  $X$  variables are called the *original variables*. Any other variable that appears in a (promise resolution) refutation is an *extension variable*.

**Definition 2 (CNF formulas under a promise).** Let  $\Lambda$  be a fixed function in  $n$  (the number of  $X$  variables) such that  $0 \leq \Lambda(n) \leq 2^n$ . The function  $\Lambda$  is called the *promise*. The set of CNF formulas under the promise  $\Lambda$  consists of all CNF formulas in the  $X$  variables that are either unsatisfiable or have more than  $\Lambda(n)$  satisfying assignments (for  $n = |X|$ ).

The refutation systems we build are sound and complete for the set of CNF formulas under a (given) promise. That is, every unsatisfiable CNF formula has a refutation in the system (this corresponds to completeness), while no CNF having  $n$  variables and more than  $\Lambda(n)$  satisfying assignments has a refutation in it (this corresponds to soundness under the promise). The soundness (under the promise) is achieved by requiring that resolution should *prove the fact that we discard the right number of assignments* (see Sect. 3.1 for details).

**Definition 3 (Assignment discarding).** *Let  $A$  be a CNF in the  $X$  variables that can contain (but not necessarily does) extension variables (that is, variables not from  $X$ ). We say that an assignment to the  $X$  variables  $\mathbf{a}$  is discarded by  $A$  if there is no extension of  $\mathbf{a}$  (to the extension variables in  $A$ ) that satisfies  $A$ .*

### 3.1 Promise Axioms

**Big promise.** We first concentrate on a promise of a *constant fraction of assignments*. Let the promise (see Definition 2) be  $\Lambda = \varepsilon \cdot 2^n$ , for a constant  $0 < \varepsilon < 1$  (we fix this  $\Lambda$  throughout this subsection), and let  $r = \lceil \log(1/\varepsilon) \rceil$  and  $t = 2^r - 1$ . Let  $C$  be a sequence of Boolean circuits  $C := (C^{(1)}, \dots, C^{(t)})$ . Assume that each  $C^{(i)}$  has  $n - r$  input bits and  $n$  output bits and computes the Boolean map  $f_i : \{0, 1\}^{n-r} \rightarrow \{0, 1\}^n$ . Assume further that the  $f_i$ 's are all injective maps and that the images of all these maps are pairwise disjoint. Denote by  $\text{Im}(f_i)$  the image of the map  $f_i$ . For simplicity, we call the union  $\bigcup_{i=1}^t \text{Im}(f_i)$  *the image of  $C$*  and denote it by  $\text{Im}(C)$ . By the definition of  $r$ , we have  $2^{n-r} \leq \varepsilon \cdot 2^n = \Lambda$ , and by the injectivity and pairwise disjointness of the images of the  $f_i$ 's we have:

$$|\text{Im}(C)| = t \cdot 2^{n-r} = (2^r - 1) \cdot 2^{n-r} = 2^n - 2^{n-r} \geq 2^n - \Lambda. \tag{1}$$

Therefore, *we can treat  $\text{Im}(C)$  as the set of all possible truth assignments for the original variables  $X$ , without losing soundness*: If  $K$  is unsatisfiable then there is no assignment in  $\text{Im}(C)$  that satisfies  $K$ ; and if  $K$  is satisfiable then according to the promise it has more than  $\Lambda$  satisfying assignments, which means that there is at least one assignment in  $\text{Im}(C)$  that satisfies  $K$ . This idea is formulated as a propositional formula, as follows:

**Definition 4 (Promise Axiom for  $\Lambda = \varepsilon \cdot 2^n$ ).** *Let the promise be  $\Lambda = \varepsilon \cdot 2^n$ , for a constant  $0 < \varepsilon < 1$ , and let  $r = \lceil \log(1/\varepsilon) \rceil$  and  $t = 2^r - 1$ . Let  $C$  be a sequence of Boolean circuits  $C := (C^{(1)}, \dots, C^{(t)})$ . Assume that each  $C^{(i)}$  has  $n - r$  input bits and  $n$  output bits and let  $\mathbf{W}_1$  and  $\mathbf{W}_2$  be two disjoint sets of  $n - r$  extension variables each. The promise axiom  $\text{PRM}_{C,\Lambda}$  is the CNF encoding of the following Boolean formula (see the encoding in the full version [4]):*

$$\left( \bigwedge_{i=1}^t (C^{(i)}(\mathbf{W}_1) \equiv C^{(i)}(\mathbf{W}_2) \rightarrow \mathbf{W}_1 \equiv \mathbf{W}_2) \wedge \bigwedge_{1 \leq i < j \leq t} C^{(i)}(\mathbf{W}_1) \not\equiv C^{(j)}(\mathbf{W}_2) \right) \longrightarrow \bigvee_{i=1}^t C^{(i)}(\mathbf{W}_1) \equiv X.$$

(The notation  $\mathbf{W}_1 \equiv \mathbf{W}_2$  means that the  $i$ th variable in  $\mathbf{W}_1$  is logically equivalent to the  $i$ th variable in  $\mathbf{W}_2$ , and similarly for  $C^{(i)}(\mathbf{W}_1) \equiv C^{(i)}(\mathbf{W}_2$ ); see the full version of this paper for more details.) The promise axiom  $\text{PRM}_{C,\Lambda}$  expresses the fact that if each circuit in  $C$  computes an injective map (this is formulated as  $\bigwedge_{i=1}^t (C^{(i)}(\mathbf{W}_1) \equiv C^{(i)}(\mathbf{W}_2) \rightarrow \mathbf{W}_1 \equiv \mathbf{W}_2)$ ), and if the images of the maps computed by each pair of circuits in  $C$  are disjoint (this is formulated as  $\bigwedge_{1 \leq i < j \leq t} C^{(i)}(\mathbf{W}_1) \not\equiv C^{(j)}(\mathbf{W}_2)$ ), then we can assume that the assignments to the original variables  $X$  are taken from the image of  $C$  (this is formulated as  $\bigvee_{i=1}^t C^{(i)}(\mathbf{W}_1) \equiv X$ ). The fact that the image of  $C$  is of size at least  $2^n - \Lambda$  is expressed (due to Eq. (II)) by the number of input bits (i.e.,  $n - r$ ) of each circuit in  $C$  and the number of circuits in  $C$  (i.e.,  $t$ ). Also note that the promise axiom is of polynomial-size as long as the circuits in  $C$  are (since  $1/\varepsilon$  is a constant).

The following claim states that the promise axioms are sound with respect to the promise  $\Lambda$  in the sense that they do not discard too many truth assignments (see the full version [4] for the proof):

*Claim.* The promise axiom  $\text{PRM}_{C,\Lambda}$  discards at most  $\Lambda$  truth assignments. That is, there are at most  $\Lambda$  distinct assignments  $\mathbf{a}$  to the  $X$  variables such that  $\text{PRM}_{C,\Lambda} \models X \not\equiv \mathbf{a}$ .

**Smaller promise.** We are also interested in formulating promise axioms for promises smaller than  $\varepsilon \cdot 2^n$ . Specifically, we are interested in the promise  $\Lambda = 2^{\delta n}$  for a constant  $0 < \delta < 1$ . For such a promise, the promise axiom is essentially similar to Definition 4, except that the number of input bits of each circuit in  $C$  needs to be modified accordingly. Due to space limitations, we do not describe the formulation of this kind of promise axiom (and refer the interested reader to the full version [4]).

### 3.2 Promise Resolution

**Definition 5 (Promise resolution).** Let  $\Lambda$  be the promise (see Definition 2) and let  $K$  be a CNF in the  $X$  variables. A promise resolution (under the promise  $\Lambda$ ) proof of the clause  $D$  from a CNF formula  $K$  is a sequence of clauses  $D_1, D_2, \dots, D_\ell$  such that: (1) Each clause  $D_j$  is either a clause of  $K$  or a clause of a promise axiom  $\text{PRM}_{C,\Lambda}$  (where  $\text{PRM}_{C,\Lambda}$  is either a big or a smaller promise axiom as defined, for instance, in Definitions 4, and  $C$  is an arbitrary sequence of circuits with the prescribed input and output number of bits) or a resolvent of two previous clauses in the sequence or a weakening of a previous clause; (2) The sequence contains (the clauses of) at most one promise axiom; (3) The last clause  $D_\ell = D$ . The size and refutations of promise resolution is defined the same as for resolution.

Note that promise resolution is a Cook-Reckhow proof system (see the introduction for a definition), in the sense that it is possible to efficiently verify whether a given CNF is an instance of the promise axiom, and hence to verify whether a sequence of clauses constitute a legitimate promise refutation. This can be done by “decoding” the CNF that encodes the promise axiom  $\text{PRM}_{C,\Lambda}$  and then checking that each circuit in  $C$  has the right number of input and output bits.

**Proposition 1.** *Let  $\Lambda$  be the promise (where  $\Lambda$  is either  $\varepsilon \cdot 2^n$  or  $2^{\delta n}$ , for  $0 < \varepsilon, \delta < 1$ ). Then, promise resolution under promise  $\Lambda$  is a sound and complete proof system for the set of CNF formulas under the promise  $\Lambda$ . In other words, every unsatisfiable CNF has a promise resolution refutation, and every CNF that has more than  $\Lambda$  satisfying assignments does not have promise resolution refutations.*

*Proof.* Completeness stems from completeness of resolution. Soundness under the promise  $\Lambda$  stems from Claim 3.1 (This claim refers to the big promise, but a similar argument holds also for the smaller promise axiom.)

The full paper [4] contains a brief discussion of the definition of promise proofs and the choice made in formulating the above promise axioms.

## 4 Big Promise – The Upper Bound

We sketch a proof showing that under the promise  $\Lambda = \varepsilon \cdot 2^n$ , for any constant  $0 < \varepsilon < 1$ , resolution can efficiently certify the unsatisfiability of all unsatisfiable 3CNF formulas. The method resembles the algorithm presented by Trevisan [7] for approximating the number of satisfying assignments of a  $k$ CNF formula.

The idea behind the refutations in this section is based on the following observation: Given an unsatisfiable 3CNF formula  $K$  and a constant  $c$ , either there are  $3(c-1)$  variables that hit<sup>2</sup> all the clauses in  $K$  or there are at least  $c$  clauses in  $K$  over  $3c$  distinct variables denoted by  $K'$  (that is, each variable in  $K'$  appears only once). In the first case, we can consider all the possible truth assignments to the  $3c$  variables inside resolution: If  $K$  is unsatisfiable then any such truth assignment yields an unsatisfiable 2CNF formula, which can be efficiently refuted in resolution. In the second case, we can make use of a promise axiom to efficiently refute  $K'$  (this set of clauses has less than  $\Lambda$  satisfying assignments, for sufficiently large  $c$ ). Specifically, in the second case, we construct a sequence of small circuits  $C$  for which any satisfying assignment for  $K'$  is *provably in resolution* (with polynomial-size proofs) outside the image of  $C$ . The following is the main result of this section:

**Theorem 1.** *Let  $0 < \varepsilon < 1$  be a constant and let  $\Lambda = \varepsilon \cdot 2^n$  be the given promise. Then every unsatisfiable 3CNF with  $n$  variables has a polynomial-size (in  $n$ ) resolution refutation under the promise  $\Lambda$ .*

This theorem is a consequence of the three lemmas that follow (their proofs appear in the full version [4]):

**Lemma 1.** *Let  $K$  be a 3CNF formula. For every integer  $c$  one of the following holds: (i) there is a set of at most  $3(c-1)$  variables that hit all the clauses in  $K$ ; or (ii) there is a sub-collection of clauses from  $K$ , denoted  $K'$ , with at least  $c$  clauses and where each variable appears only once in  $K'$ .*

<sup>2</sup> A set of variables  $S$  that hit all the clauses in a CNF formula  $K$  is a set of variables for which every clause in  $K$  contains some variable from  $S$ .

If case (i) of the prior lemma holds, then the following lemma suffices to efficiently refute the 3CNF:

**Lemma 2.** *Let  $c$  be constant and  $K$  be an unsatisfiable 3CNF formula in the  $X$  variables (where  $n = |X|$ ). Assume that there is a set  $S \subseteq X$  of at most  $3(c - 1)$  variables that hit all the clauses in  $K$ . Then, there is a polynomial-size (in  $n$ ) resolution refutation of  $K$ .*

If case (ii) in Lemma 1 holds, then it suffices to show that resolution under a big promise can efficiently refute any 3CNF formula  $T$  with a constant number of clauses (for a sufficiently large constant), where *each variable in  $T$  occurs only once* (such a  $T$  is of course satisfiable, but it has less than an  $\varepsilon$  fraction of satisfying assignments for a sufficiently large number of clauses):

**Lemma 3.** *Fix the constant  $c = 3\lceil \log_{7/8}(\varepsilon/2) \rceil$ . Let  $\Lambda = \varepsilon \cdot 2^n$ , where  $0 < \varepsilon < 1$  is a constant and  $n$  is sufficiently large. Assume that  $T$  is a 3CNF with  $c/3$  clauses (and  $c$  variables) over the  $X$  variables, where each variable in  $T$  occurs only once inside  $T$ . Then, there is a polynomial-size resolution refutation of  $T$  under the promise  $\Lambda$ .*

The proof of Lemma 3 consists of constructing a sequence of polynomial-size circuits  $C$  (where the parameters of the circuits in  $C$  are taken from Definition 4 that is,  $r = \lceil \log(1/\varepsilon) \rceil$  and  $t = 2^r - 1$ ), such that: (i) resolution can efficiently prove the injectivity and the pairwise disjointness of the images of the circuits in  $C$ ; and (ii) there is a polynomial-size refutation of  $T$  and  $\text{PRM}_{\Lambda, C}$ . In other words, there is a polynomial-size derivation of the empty clause from the clauses of both  $T$  and  $\text{PRM}_{\Lambda, C}$ .

## 5 Smaller Promise – The Lower Bound

In this section, we state an exponential lower bound on the size of resolution refutations under the promise  $2^{\delta n}$ , for any constant  $0 \leq \delta \leq 1$ . The lower bound applies to random 3CNF formulas with  $o(n^{3/2})$  number of clauses (where  $n$  is the number of variables in the 3CNF). This lower bound matches the known lower bound on resolution refutation-size for random 3CNF formulas (without any promise). Basically, the proof strategy of our lower bound is similar to that of [2], except that we need to take care that every step in the proof works with the augmented (smaller) promise axiom.

The lower bound is somewhat stronger than described above in two respects. First, we show that restricting the set of all  $2^n$  truth assignments to *any* smaller set (not just those sets defined by small circuits) that consists of  $2^n - 2^{\delta n}$  assignments (for any constant  $0 \leq \delta \leq 1$ ), does not give resolution any advantage in the average-case. One can think of such a restriction as modifying the semantic implication relation  $\models$  to take into account only assignments from some prescribed set of assignments  $S$ , such that  $|S| = 2^n - 2^{\delta n}$  (in other words, for two formulas  $A, B$ , we have that  $A \models B$  under the restriction to  $S$  iff any truth

assignment from  $S$  that satisfies  $A$  also satisfies  $B$ ). Formally, this means that the lower bound does not use the fact that the restricted domain of size  $2^n - 2^{\delta n}$  is defined by a sequence  $C$  of polynomial-size circuits (nor the fact that the circuits in  $C$  ought to have polynomial-size resolution proofs of their injectivity and pairwise disjointness). Second, we could allow for a promise that is bigger than  $2^{\delta n}$ , and in particular for a promise of  $2^{n(1-1/n^{1-\xi})} = 2^n/2^{n^\xi}$ , for some constant  $0 < \xi < 1$ .

The following defines the usual *average-case* setting of 3CNF formulas:

**Definition 6 (Random 3CNF formulas).** *For a 3CNF formula  $K$  with  $n$  variables  $X$  and  $\beta \cdot n$  clauses, we say that  $\beta$  is the density of  $K$ . A random 3CNF formula on  $n$  variables and density  $\beta$  is defined by picking  $\beta \cdot n$  clauses from the set of all  $2^3 \cdot \binom{n}{3}$  clauses, independently and indistinguishably distributed, with repetitions.*

Finally, the next theorem gives our lower bound. A complete proof appears in the full version [4].

**Theorem 2.** *Let  $0 < \delta < 1$  and  $0 < \epsilon < 1/2$ . With high probability a random 3CNF formula with  $\beta = n^{1/2-\epsilon}$  requires a size  $\exp(\Omega(\beta^{-4/(1-\epsilon)} \cdot n))$  resolution refutation under the promise  $\Lambda = 2^{\delta n}$ .*

## Acknowledgments

The second author is indebted to Ran Raz for very helpful conversations that led to the present paper. We also wish to thank Jan Krajíček for commenting on an earlier version of this paper and Eli Ben-Sasson and Amnon Ta-Shma for useful correspondence and conversations.

## References

1. Beame, P., Karp, R., Pitassi, T., Saks, M.: The efficiency of resolution and Davis-Putnam procedures. *SIAM J. Comput.* 31(4), 1048–1075 (2002)
2. Ben-Sasson, E., Wigderson, A.: Short proofs are narrow—resolution made simple. *J. ACM* 48(2), 149–169 (2001)
3. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic* 44(1), 36–50 (1979)
4. Dershowitz, N., Tzameret, I.: Complexity of propositional proofs under a promise (full version) (2007) <http://www.cs.tau.ac.il/~tzameret/PromiseProofs.pdf>
5. Feige, U., Kim, J., Ofek, E.: Witnesses for non-satisfiability of dense random 3CNF formulas. In: Proc. 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 497–508 (October 2006)
6. Hirsch, E.: A fast deterministic algorithm for formulas that have many satisfying assignments. *Logic Journal of the IGPL* 6(1), 59–71 (1998)
7. Trevisan, L.: A note on approximate counting for  $k$ -DNF. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) *RANDOM 2004* and *APPROX 2004*. LNCS, vol. 3122, pp. 417–426. Springer, Heidelberg (2004)

# Deterministic History-Independent Strategies for Storing Information on Write-Once Memories<sup>\*</sup>

Tal Moran, Moni Naor<sup>\*\*</sup>, and Gil Segev

Department of Computer Science and Applied Mathematics,  
Weizmann Institute of Science, Rehovot 76100, Israel  
{tal.moran, moni.naor, gil.segev}@weizmann.ac.il

**Abstract.** Motivated by the challenging task of designing “secure” vote storage mechanisms, we deal with information storage mechanisms that operate in extremely hostile environments. In such environments, the majority of existing techniques for information storage and for security are susceptible to powerful adversarial attacks. In this setting, we propose a mechanism for storing a set of at most  $K$  elements from a large universe of size  $N$  on write-once memories in a manner that does not reveal the insertion order of the elements. Whereas previously known constructions were either inefficient (required  $\Theta(K^2)$  memory), randomized, or employed cryptographic techniques which are unlikely to be available in hostile environments, we eliminate each of these undesirable properties. The total amount of memory used by the mechanism is linear in the number of stored elements and poly-logarithmic in the size of the universe of elements.

In addition, we consider one of the classical distributed computing problems: Conflict resolution in multiple-access channels. By establishing a tight connection with the basic building block of our mechanism, we construct the first deterministic and non-adaptive conflict resolution algorithm whose running time is optimal up to poly-logarithmic factors.

## 1 Introduction

We consider the abstract problem of storing a set of at most  $K$  elements taken from a large universe of size  $N$ , while minimizing the total amount of allocated memory. We design a storage mechanism which is deterministic, history-independent, and tamper-evident. Our mechanism supports insert operations, membership queries, and enumeration of all stored elements. Whereas previously known constructions were either inefficient, randomized, or employed cryptographic techniques that require secure key storage, we make a concentrated effort to eliminate these undesirable properties.

---

<sup>\*</sup> Due to space limitations we refer the reader to a longer version available at <http://www.wisdom.weizmann.ac.il/~naor>. Research supported in part by a grant from the Israel Science Foundation.

<sup>\*\*</sup> Incumbent of the Judith Kleeman Professorial Chair.



Our motivation emerges from the task of designing vote storage mechanisms<sup>[1]</sup>, recently studied by Molnar, Kohno, Sastry and Wagner [8]. They described the desirable security goals of such mechanisms and suggested simple constructions. Without a “secure” vote storage mechanism, an adversary may be able to undetectably tamper with the voting records or compromise voter privacy. A typical threat is a corrupt poll worker who has complete access to the vote storage mechanism at some point during or after the election process. In order to prevent the adversary from modifying the stored votes, a vote storage mechanism should be *tamper-evident*. Moreover, as the order in which votes are cast must be hidden to protect the privacy of the voters, a vote storage mechanism should be *history-independent* as well. We refer the reader to [8] for a detailed list of the security goals of vote storage mechanisms.

In this paper we deal with the design of information storage mechanisms that operate in extremely hostile environments. In such environments, the majority of existing techniques for information storage and for security are susceptible to powerful adversarial attacks. In order to prevent such attacks, we study storage mechanisms with special properties.

**Deterministic strategies.** Randomization is an important ingredient in the design of efficient systems. However, for systems that operate in hostile environments, randomization can assist the adversary in attacking the system. First, as sources of random bits are typically obtained from the environment, it is quite possible that the adversary can corrupt these sources. In such cases, we usually have no guarantees on the expected behavior of the system. Second, even when truly random bits are available, these bits may be revealed to the adversary in advance, and serve as a crucial tool in the attack. Third, a randomized storage strategy may enable a subliminal channel: As multiple valid representations for the same abstract state exist, a malicious storage mechanism can secretly embed information into the stored data by choosing one of these representations. Applications such as voting protocols may run in completely untrusted environments. In such cases, deterministic strategies have invaluable security benefits.

**History-independence.** Many systems give away much more information than they were intended to. When designing a data structure whose memory representation may be revealed, we would like to ensure that an adversary will not be able to infer information that is not available through the system’s legitimate interface. Computer science is rich with tales of cases where this was not done, such as files containing information whose creators assumed had been erased, only to be revealed later in embarrassing circumstances. Very informally, we consider a period of activity after which an adversary gains complete control over the data structure. In particular, the memory representation of the data is revealed to the adversary. The data structure is history-independent if the adversary will not be

---

<sup>1</sup> We note that for vote storage mechanisms it is sufficient to support only insert operations and enumeration of all stored elements. Our mechanism supports (efficient) membership queries as well, although we did not set out to implement this property.

able to deduce any more about the sequence of operations that led to the current content than the content itself yields.

**Tamper-evident write-once storage.** A data structure is tamper-evident if any unauthorized modification of its content can be detected. Tamper-evidence is usually provided by a mixture of physical assumptions (such as secure processors) and cryptographic tools (such as signature schemes). Unfortunately, the majority of cryptographic tools require secure key storage, which is unlikely to be available in a hostile environment. Our construction follows the approach of Molnar et al. [8], who exploited the properties of write-once memories to provide tamper-evident storage. They introduced an encoding scheme in which flipping some of the bits of any valid codeword from 0 to 1 will never lead to another valid codeword<sup>2</sup>. In the voting scenario, this prevents any modification to the stored ballots after the polls close, and prevents poll workers from tampering with the content of the data structure while the storage device is in transit. This approach does not require any cryptographic tools or computational assumptions, which makes it very suitable for the setting of hostile environments. The additional memory allocation required by the encoding is only logarithmic in the size of the stored data, and can be handled independently of the storage strategy. For simplicity of presentation, we ignore the encoding procedure, and refer the reader’s attention to the fact that our storage strategy is indeed write-once (i.e., the memory is initialized to the all 0’s state, and the only operation allowed is flipping bits from 0 to 1).

**Conflict resolution.** In this paper we also address a seemingly unrelated problem: *conflict resolution in multiple-access channels*. A fundamental problem of distributed computing is to resolve conflicts that arise when several stations transmit simultaneously over a single channel. A conflict resolution algorithm schedules retransmissions, such that each of the conflicting stations eventually transmits singly to the channel. Such an algorithm is *non-adaptive* if the choice of the transmitting stations in each step does not depend on information gathered from previous steps. The efficiency measure for conflict resolution algorithms is the total number of steps it takes to resolve conflicts in the worst case<sup>3</sup>.

We consider the standard model in which  $N$  stations are tapped into a single channel, and there are at most  $K$  conflicting stations. In 1985, Komlós and Greenberg [6] provided a *non-constructive* proof for the existence of a deterministic and non-adaptive algorithm that resolves conflicts in  $O(K \log(N/K))$  steps. However, no explicit algorithm with a similar performance guarantee was known. By establishing a tight connection between this problem and the basic building block of our storage mechanism, we construct the first efficient deterministic and non-adaptive conflict resolution algorithm.

---

<sup>2</sup> Consider the encoding  $E(x) = x \parallel \text{wt}(\bar{x})_2$ , obtained by concatenating the string  $x$  with the binary representation of the Hamming weight of its complement. Flipping any bit of  $x$  from 0 to 1 decreases  $\text{wt}(\bar{x})_2$ , and requires flipping at least one bit of  $\text{wt}(\bar{x})_2$  from 1 to 0.

<sup>3</sup> Worst case refers to the maximum over all possible sets of conflicting stations.

## 1.1 Our Contributions

We construct a deterministic mechanism for storing a set of at most  $K$  elements on write-once memories. The elements are given one at a time, and stored in a manner that does not reveal the insertion order. Our mechanism is immune to a large class of attacks that made previous constructions unsuitable for extremely hostile environments. Whereas previous constructions were either inefficient (required  $\Theta(K^2)$  memory), randomized, or employed cryptographic techniques that require secure key storage, we eliminate each of these undesirable properties. Our main result is the following<sup>4</sup>:

**Theorem 1.** *There exists an explicit, deterministic, history-independent, and write-once mechanism for storing a set of at most  $K$  elements from a universe of size  $N$ , such that:*

1. *The total amount of allocated memory is  $O(K \cdot \text{polylog}(N))$ .*
2. *The amortized insertion time is  $O(\text{polylog}(N))$ .*
3. *The worst-case look-up time is  $O(\text{polylog}(N))$ .*

In addition, our construction yields a non-constructive proof for the existence of the following storage mechanism:

**Theorem 2.** *There exists a deterministic, history-independent, and write-once mechanism for storing a set of at most  $K$  elements from a universe of size  $N$ , such that:*

1. *The total amount of allocated memory is  $O(K \log(N/K))$ .*
2. *The amortized insertion time is  $O(\log(N/K))$ .*
3. *The worst-case look-up time is  $O(\log N \cdot \log K)$ .*

Finally, by adapting our technique to the setting of conflict resolution, we devise the first efficient deterministic and non-adaptive algorithm for this problem. The number of steps required by our algorithm to resolve conflicts matches the non-explicit upper bound of Komlós and Greenberg [6] up to poly-logarithmic factors. In the full version of this paper we prove the following theorem:

**Theorem 3.** *For every  $N$  and  $K$  there exists an explicit, deterministic, and non-adaptive algorithm that resolves any  $K$  conflicts among  $N$  stations in  $O(K \cdot \text{polylog}(N))$  steps.*

**Paper organization.** The rest of the paper is organized as follows. In the remainder of this section we review related work and essential definitions. In Section 2 we present our security goals and threat model. In Section 3 we present an overview of our storage mechanism, which is then described in Section 4.

---

<sup>4</sup> For simplicity, throughout the paper we refer to the amount of allocated memory as the number of allocated memory *words*, each of length  $\log N$  bits. We assume that the mechanism can read and write a memory word in constant time.

## 1.2 Related Work

The problem of constructing history-independent data structures was first formally considered by Micciancio [7], who devised a variant of 2–3 trees that satisfies a property of this nature. Micciancio considered a rather weak notion of history-independence, which required only that the *shape* of the trees does not leak information. We follow Naor and Teague [9] and consider a stronger notion – data structures whose *memory representation* does not leak information. Naor and Teague focused on dictionaries, and constructed very efficient hash tables in which the cost of each operation is constant.

In the context of write-once memories, Rivest and Shamir [10] initiated the study of codes for write-once memory, by demonstrating that such memories can be “rewritten” to a surprising degree. Irani, Naor and Rubinfeld [5] explored the time and space complexity of computation using write-once memories, i.e., whether “a pen is much worse than a pencil”. They proved that a Turing machine with write-once polynomial space decides exactly the class of languages P.

**History-independence on write-once memories.** Molnar et al. [8] studied the task of designing a vote storage mechanism, and suggested constructions of history-independent storage mechanisms on write-once memories. Among their suggestions is a deterministic mechanism based on an observation of Naor and Teague [9], stating that one possible way of ensuring that the memory representation is determined by the content of a data structure is to store the elements in lexicographical order. This way, any set of elements has a single canonical representation, regardless of the insertion order of its elements. When dealing with write-once media, however, we cannot sort in-place when a new element is inserted. Instead, on every insertion, we compute the sorted list that includes the new element, copy the contents of this list to the next available memory position, and erase the previous list. We refer to this solution as a *copy-over list*, as suggested by Molnar et al. [8]. The main disadvantage of copy-over lists is that any insertion requires copying the entire list. Therefore, storing  $K$  elements requires  $\Theta(K^2)$  memory<sup>5</sup>.

In an attempt to improve the amount of allocated memory, Molnar et al. suggested using a hash table in which each entry is stored as a separate copy-over list. The copy-over lists are necessary when several elements are mapped to the same entry. However, with a fixed hash function the worst-case behavior of the table is very poor, and therefore the hash function must be randomly chosen and hidden from the adversary. Given the hash function, the mechanism is deterministic and we refer to such a strategy as an *off-line* randomized strategy. For instance, the mechanism may choose a pseudo-random function as its hash function. However, this approach is not suitable for hostile environments, where secure storage for the key of the hash function is not available.

---

<sup>5</sup> When dealing with a small universe, a better solution is to pre-allocate memory to store a bounded unary counter for each element. However, this may not be suitable for vote storage in cases where write-in candidates are allowed (as common in the U.S.) or when votes are subsets or rankings (as common in many countries).

Molnar et al. also showed that an *on-line* randomized strategy can significantly improve the amount of allocated memory. A simple solution is to allocate an array of  $2K$  entries, and insert an element by randomly probing the array until an empty entry is found. However, as mentioned earlier, such a strategy may enable subliminal channels: a malicious storage mechanism can secretly embed information into the stored data by choosing among the multiple valid representations of the same data.

**Tamper-evidence without write-once memories.** Whereas the constructions of Molnar et al. achieved tamper-evidence by exploiting the properties of write-once memories, a different approach was taken by Bethencourt, Boneh and Waters [2]. They designed a history-independent tamper-evident storage mechanism by constructing a signature scheme which signs sets of elements such that the order in which elements were added cannot be determined, and elements cannot be deleted from the set. Even though their solution uses only  $O(K)$  memory to store  $K$  elements, it is randomized and requires secure storage for cryptographic keys (as well as computational assumptions).

### 1.3 Formal Definitions

A data structure is defined by a list of operations. We construct a data structure that supports the following operations:

1. **Insert**( $x$ ) - stores the element  $x$ .
2. **Seal**() - finalizes the data structure (after this operation no **Insert** operations are allowed).
3. **LookUp**( $x$ ) - outputs **FOUND** if and only if  $x$  has already been stored.
4. **RetrieveAll**() - outputs all stored elements.

We say that two sequences of operations,  $S_1$  and  $S_2$ , yield the same content if for all suffixes  $T$ , the results returned by  $T$  when the prefix is  $S_1$  are identical to those returned by  $T$  when the prefix is  $S_2$ .

**Definition 4.** *A deterministic data structure is history-independent if any two sequences of operations that yield the same content induce the same memory representation.*

In our scenario, two sequences of operations yield the same content if and only if the corresponding sets of stored elements are identical. The above definition is a simplification of the one suggested by Naor and Teague [9], when dealing only with deterministic data structures. Naor and Teague also considered a stronger definition, in which the adversary gains control *periodically*, and obtains the current memory representation at several points along the sequence of operations. This definition has also been studied by Hartline et al. [4] and by Buchbinder and Petrank [3]. Since we deal only with deterministic data structures, in our setting the definitions are equivalent.

## 2 Security Goals and Threat Model

Our approach in defining the security goals and threat model is motivated by the possible attacks on an electronic voting system. To make the discussion clearer, we frame the threat model in terms of a vote storage mechanism. In an actual voting scenario, we think of ballot casting as an **Insert** operation. In the most trivial case, the element inserted is simply the chosen candidate’s name. In more complex voting schemes, the inserted element may be a ranking of the candidates, an encrypted form of the ballot, or a combination of multiple choices. These possibilities are the reason for viewing the “universe of elements” as large, while the actual number of elements inserted is small (at most the number of voters). Once the voting is complete (e.g., the polls close), the **Seal** operation is performed. The purpose is to safeguard the ballots during transport (and for possible auditing). Finally, to count the votes, the **RetrieveAll** operation is performed. Note that in a standard voting scenario, **Lookup** is not needed – we provide it here only for completeness. The main security goals we would like our storage mechanism to achieve are the following<sup>6</sup>:

1. **Tamper-evidence:** Any modification or deletion of votes after they were cast must be detected.
2. **Privacy:** No information about the order in which votes were cast should be revealed.
3. **Robustness:** No adversary should be able to cause the election process to fail.

We consider extremely powerful adversaries: Computationally unbounded adversaries that can adaptively corrupt any number of voters (i.e., the adversary can choose to perform arbitrary **Insert** operations at arbitrary points in time). In addition, we assume the adversary can modify the mechanism’s program code *before* the election process begins, as long as the resulting code creates a “correct-looking” data structure (so that for any sequence of operations, a tester cannot distinguish between the memory representation output by the adversary’s program and the memory representation output by a correct program). In our case, since our mechanism is deterministic, “correct-looking” is equivalent to actually being correct (whereas a randomized mechanism can enable in such a case a subliminal channel, as mentioned earlier).

The extent to which each of the above goals can be achieved depends on the assumed access that the adversary has to the mechanism. More specifically, we consider differing levels of adversarial access:

**The full-access adversary.** In the worst case, the adversary gains *complete* control of the mechanism at some point in time. After this point, the adversary has full read-write access both to the stored data and to the program code. The only limitation on the adversary’s capabilities is our physical write-once

---

<sup>6</sup> For simplicity we focus on the main and most relevant security goals. We refer the reader to [8] for a more detailed list.

assumption: The adversary cannot flip bits from 1 to 0 in the stored data. Such an adversary can always erase existing information (by setting all bits to 1), thus causing the election to fail. Such an adversary can also change the program code, so it can both record and change any votes cast. The best we can hope for in this case is a guarantee about operations that took place before the attack. The adversary should not be able to *undetectedly* modify or delete votes that were previously cast, or to gain information about the order in which these votes were cast (beyond the order of the votes cast by corrupt voters).

**The limited-access adversary.** A slightly more optimistic case is when the adversary has write access only to the stored data, but not to the program code. Such an assumption may be plausible if the program code is stored in a read-only memory, or if the adversary gains access to the machine only after the election process has concluded. In this case, the adversary can still cause the election to fail (e.g., by erasing all stored data). However, the adversary should not be able to undetectably modify or delete any vote cast (except by corrupt voters). Moreover, the adversary should not gain any information about the order in which votes were cast beyond what would be gained by executing `RetrieveAll()` at any point in which the adversary has access to the mechanism.

**The read-only adversary.** In a best case scenario, the adversary has read-only access to the mechanism. This may occur if the adversary's access to the machine is through a programming bug, for instance. In this case, in addition to the security guarantees that we expect from the limited-access adversary, we would like to guarantee robustness as well: The adversary should not be able to cause the election to fail. This guarantee is not trivial, since the read-only adversary can still adaptively corrupt any number of voters.

**Maliciously adding votes.** An adversary with write access to the storage memory can always add votes (by simply executing the `Insert` operation). To detect such tampering, we combine two strategies: The first is adding a `Seal` operation to the mechanism, after which the `Insert` operation is no longer allowed. This will detect any tampering that occurs after the election process has concluded (as suggested by Molnar et al. [8]). The second is maintaining an independent count of the number of votes cast (e.g., voter lists maintained by election officials). Since our construction guarantees that votes cannot be deleted, if an adversary adds bogus votes then the counts will not match up. Note that an independent count prevents adversaries from adding votes even if they gain access to the mechanism *during* the election process.

### 3 Overview of the Construction

Our construction relies on the fundamental technique of storing elements in a hash table and resolving collisions separately in each entry of the table. More specifically, our storage mechanism incorporates two “strategies”: a *global strategy* that maps elements to the entries of the table, and a *local strategy* that resolves collisions that occur when several elements are mapped to the same entry.

As long as both strategies are deterministic, history-independent and write-once, the entire storage mechanism will also share these properties.

**The local strategy.** We resolve collisions by storing the elements mapped to each entry of the table in a separate copy-over list. Copy-over lists were introduced by Molnar et al. [8], and are based on an observation by Naor and Teague [9], stating that one possible way of ensuring that the memory representation is determined by the content of a data structure is to store the elements in lexicographical order. When dealing with write-once media, however, we cannot sort in-place when a new element is inserted. Instead, on every insertion, we compute the sorted list that includes the new element, copy the contents of this list to the next available memory position, and erase the previous list (by setting all the bits to 1). Note that storing  $K$  elements in a copy-over list requires  $\Theta(K^2)$  memory, and therefore is reasonable only for small values of  $K$ .

**The global strategy.** Our goal is to establish a *deterministic* strategy for mapping elements to the entries of the table. However, for any fixed hash function, the inserted elements can be chosen such that the load in at least one of the entries will be too high to be efficiently handled by our local strategy. Therefore, in order to ensure that the number of elements mapped to each entry remains relatively small (in the worst case), we must apply a more sophisticated strategy.

Our global strategy stores the elements in a *sequence* of tables, where each table enables us to store a fraction of the elements. Each element is first inserted into *several* entries of the first table. When an entry overflows (i.e., more than some pre-determined number of elements are inserted into it), the entry is “permanently deleted”. In this case, any elements that were stored in this entry and are not stored elsewhere in the table are inserted into the next table in a similar manner. Thus, we are interested in finding a sequence of functions that map the universe of elements to the entries of the tables, such that the total number of tables, the size of each table, and the number of collisions are minimized. We view such functions as bipartite graphs  $G = (L, R, E)$ , where the set of vertices on the left,  $L$ , is identified with the universe of elements, and the vertices on the right,  $R$ , are identified with the entries of a table. Given a set of elements  $S \subseteq L$  to store, the number of elements mapped to each table entry  $y \in R$  is the number of neighbors that  $y$  has from the set  $S$ . We would like the set  $S \subseteq L$  to have as few as possible overflowing entries, i.e., as few as possible vertices  $y \in R$  with many neighbors in  $S$ .

More specifically, we are interested in bipartite graphs  $G = (L, R, E)$  with the following property: Every set  $S \subseteq L$  of size at most  $K$  contains “many” vertices with low-degree neighbors. We refer to such graphs as *bounded-neighbor expanders* [7]. Our global strategy will map all the elements in  $S$  which have a low-degree neighbor to those neighbors, and this guarantees that the table entries corresponding to those neighbors will not overflow at any stage. However, not every element in  $S$  will have a low-degree neighbor. For this reason, we use a

---

<sup>7</sup> The definition is motivated by the notion of bipartite unique-neighbor expanders presented by Alon and Capalbo [11].



sequence of bipartite graphs, all sharing the same left set  $L$ . Each graph will enable us to store a fraction of the elements in  $S$ . Formally, we define:

**Definition 5.** Let  $G = (L, R, E)$  be a bipartite graph. We say that a vertex  $x \in L$  has an  $\ell$ -degree neighbor with respect to  $S \subseteq L$ , if it has a neighbor  $y \in R$  with no more than  $\ell$  incoming edges from  $S$ .

**Definition 6.** A bipartite graph  $G = (L, R, E)$  is a  $(K, \alpha, \ell)$ -bounded-neighbor expander, if every  $S \subseteq L$  of size  $K$  contains at least  $\alpha|S|$  vertices that have an  $\ell$ -degree neighbor with respect to  $S$ .

## 4 The Construction

Let  $G_0, \dots, G_t$  denote a sequence of bounded-neighbor expanders  $G_i = (L = [N], R_i, E_i)$  with left-degree  $D_i$  (we assume that all the vertices on the left side have the same degree). The graphs are constructed such that:

- $G_0$  is a  $(K_0 = K, \alpha_0, \ell_0)$ -bounded-neighbor expander, for some  $\alpha_0$  and  $\ell_0$ .
- For every  $1 \leq i \leq t$ ,  $G_i$  is a  $(K_i, \alpha_i, \ell_i)$ -bounded-neighbor expander, for some  $\alpha_i$  and  $\ell_i$ , where  $K_i = (1 - \alpha_{i-1})K_{i-1}$ .

As described in Section 3, the elements are stored in a sequence of tables,  $T_0, \dots, T_t$ . Each table  $T_i$  is identified with the right set  $R_i$  of the bipartite graph  $G_i$ , and contains  $|R_i|$  entries denoted by  $T_i[1], \dots, T_i[|R_i|]$ . The elements are mapped to the entries of the tables and are stored there using a separate copy-over list at each entry. The copy-over list at each entry of table  $T_i$  will store at most  $\ell_i$  elements. We denote by  $|T_i[y]|$  the number of elements stored in the copy-over list  $T_i[y]$ , and use the notation  $T_i[y] = *$  to indicate that the copy-over list  $T_i[y]$  overflowed and was permanently deleted.

In order to insert or look-up an element  $x$ , we execute  $\text{Insert}(x, T_0)$  or  $\text{LookUp}(x, T_0)$ , respectively. The  $\text{Seal}()$  operation is performed as in [8] by using the encoding discussed in the introduction. The operations  $\text{Insert}(x, T_i)$ ,  $\text{LookUp}(x, T_i)$ , and  $\text{RetrieveAll}()$  are described in Figure 1.

**Lemma 7.** For every set  $S \subseteq [N]$  of size at most  $K$ , any insertion order of its elements induces the same memory representation.

**Lemma 8.** For every set  $S \subseteq [N]$  of size at most  $K$ , for every insertion order of its elements, and for every  $0 \leq i \leq t$ , the number of  $\text{Insert}(\cdot, T_i)$  calls is at most  $K_i$ .

**Lemma 9.** The storage mechanism has the following properties:

1. The total amount of allocated memory is at most  $\sum_{i=0}^t |R_i| \cdot \ell_i^2$ .
2. The amortized insertion time is at most  $\frac{1}{K} \cdot \left( \sum_{i=0}^t |R_i| \cdot \ell_i^2 \right)$ .
3. The worst-case look-up time is at most  $\sum_{i=0}^t D_i \cdot \ell_i^2$ .

```

Insert( $x, T_i$ ):
1: for all neighbors  $y$  of  $x$  in the graph  $G_i$  do
2:   if  $T_i[y] = *$  then
3:     Continue to the next neighbor of  $x$ 
4:   else if  $|T_i[y]| < \ell_i$  then
5:     Store  $x$  in the copy-over list  $T_i[y]$ 
6:   else
7:     for all  $x'$  in  $T_i[y]$  such that  $x'$  does not appear in any other list in  $T_i$  do
8:       Execute Insert( $x', T_{i+1}$ )
9:     Set  $T_i[y] \leftarrow *$  // erase the memory blocks of  $T_i[y]$ 
10: if  $x$  was not stored in any copy-over list in the previous step then
11:   Execute Insert( $x, T_{i+1}$ )

LookUp( $x, T_i$ ):
1: for all neighbors  $y$  of  $x$  in the graph  $G_i$  do
2:   if  $x$  is stored in the copy-over list  $T_i[y]$  then
3:     return FOUND and halt
4: if  $x$  was not found in a previous step and  $i = t$  then
5:   return NOT FOUND
6: else
7:   return LookUp( $x, T_{i+1}$ )

RetrieveAll():
1: for all tables  $T_i$  do
2:   for all copy-over lists  $T_i[y]$  do
3:     if  $T_i[y] \neq *$  then
4:       Output all elements of  $T_i[y]$  that have not yet been output
    
```

Fig. 1. The Insert, LookUp, and RetrieveAll operations

Theorems 11 and 12 now follow by instantiating the mechanism with the constructions of bounded-neighbor expanders stated below in Theorems 10 and 11, respectively. Due to space limitations, the proofs of these theorems appear in the full version of this paper.

**Theorem 10.** *For every  $n, k$  and constant  $\epsilon > 0$ , there exists an efficiently computable  $(K = 2^k, \alpha, 1/\alpha)$ -bounded-neighbor expander  $G = (L, R, E)$ , with  $|L| = N = 2^n$ ,  $|R| = \Theta(K/\log^3(N))$ , left-degree  $D = \text{polylog}(N)$ , and  $\alpha = \frac{(1-\epsilon)|R|}{2DK}$ .*

**Theorem 11.** *For every  $N$  and  $K$ , there exists a  $(K, 1/2, 1)$ -bounded-neighbor expander  $G = (L, R, E)$ , with  $|L| = N$ ,  $|R| = O(K \log(N/K))$  and left-degree  $D = O(\log(N/K))$ .*

## 5 Concluding Remarks

**Non-amortized insertion time.** The amortized insertion time of our storage mechanism is at most poly-logarithmic. However, the worst-case insertion time

may be larger, since an insertion may have a cascading effect. In some cases, this might enable a side-channel attack in which the adversary exploits the insertion times in order to obtain information on the order in which elements were inserted. We note that if multiple writes are allowed, then by combining our global strategy with the hashing method of Naor and Teague [9], we can achieve a poly-logarithmic worst-case insertion time, as well as linear memory allocation. Whether this is possible using write-once memory remains an open problem.

**Bounded-neighbor expanders.** The explicit bounded-neighbor expander construction stated in Theorem 10 does not achieve the parameters that one can hope for according to Theorem 11. It would be interesting to improve our explicit construction, as any such improvement will in turn lead to a more efficient instantiation of our storage mechanism.

**Memory lower bound.** The total amount of allocated *bits* required by our storage mechanism stated in Theorem 2 is  $O(K \log(N) \log(N/K))$ . This leaves a gap between the optimal construction using multiple-writes (that requires only  $O(K \log(N/K))$  bits) and our construction using write-once memory. An interesting open question is whether this gap is an unavoidable consequence of using write-once memory. This can be alternatively formulated as follows: Find the minimal integer  $m$  such that any set  $S$  of size at most  $K$  can be mapped to a binary vector  $V_S \in \{0, 1\}^m$ , with the property that  $V_{S_1} \subseteq V_{S_2}$  whenever  $S_1 \subseteq S_2$ .

## Acknowledgment

We thank Ronen Gradwohl for many useful discussions and suggestions.

## References

1. Alon, N., Capalbo, M.R.: Explicit unique-neighbor expanders. In: 43rd FOCS, pp. 73–79 (2002)
2. Bethencourt, J., Boneh, D., Waters, B.: Cryptographic methods for storing ballots on a voting machine. In: 14th NDSS, pp. 209–222 (2007)
3. Buchbinder, N., Petrank, E.: Lower and upper bounds on obtaining history-independence. *Information and Computation* 204(2), 291–337 (2006)
4. Hartline, J.D., Hong, E.S., Mohr, A.E., Pentney, W.R., Rocke, E.: Characterizing history independent data structures. *Algorithmica* 42(1), 57–74 (2005)
5. Irani, S., Naor, M., Rubinfeld, R.: On the time and space complexity of computation using write-once memory -or- Is pen really much worse than pencil? *Mathematical Systems Theory* 25(2), 141–159 (1992)
6. Komlós, J., Greenberg, A.G.: An asymptotically fast nonadaptive algorithm for conflict resolution in multiple-access channels. *IEEE Transactions on Information Theory* 31(2), 302–306 (1985)
7. Micciancio, D.: Oblivious data structures: Applications to cryptography. In: 29th STOC, pp. 456–464 (1997)

8. Molnar, D., Kohno, T., Sastry, N., Wagner, D.: Tamper-evident, history-independent, subliminal-free data structures on PROM storage -or- How to store ballots on a voting machine. In: IEEE S&P, pp. 365–370. IEEE Computer Society Press, Los Alamitos (2006)
9. Naor, M., Teague, V.: Anti-persistence: History independent data structures. In: 33rd STOC, pp. 492–501 (2001)
10. Rivest, R.L., Shamir, A.: How to reuse a write-once memory. *Information and Control* 55(1-3), 1–19 (1982)

# Trading Static for Adaptive Security in Universally Composable Zero-Knowledge

Aggelos Kiayias\* and Hong-Sheng Zhou\*

Computer Science and Engineering  
University of Connecticut  
Storrs, CT, USA  
{aggelos,hszhou}@cse.uconn.edu

**Abstract.** Adaptive security, while more realistic as an adversarial model, is typically much harder to achieve compared to static security in cryptographic protocol design. Universal composition (UC) provides a very attractive framework for the modular design of cryptographic protocols that captures both static and adaptive security formulations. In the UC framework, one can design protocols in hybrid worlds that allow access to idealized functionalities and then apply the universal composition theorem to obtain more concrete protocol instances. The zero-knowledge (ZK) ideal functionality is one of the most useful sub-protocols in modular cryptographic design. Given an adaptively secure protocol in the ideal ZK-hybrid-world do we always need an adaptively secure realization of the ZK functionality in order to preserve adaptive security under composition? In this work, perhaps surprisingly, we find that this is not so and in fact there are useful protocol instances that we can “trade static security for adaptive security.”

We investigate the above setting, by introducing a weakened ZK ideal functionality, called the ideal leaking-zero-knowledge functionality (LZK) that leaks some information about the witness to the adversary in a certain prescribed way. We show that while LZK is interchangeable to ZK against static adversaries, ZK is more stringent when adaptive adversaries are considered. We then proceed to characterize a class of protocols in the hybrid-ZK-world that can be “transported” to the LZK-hybrid-world without forfeiting their security against adaptive adversaries. Our results demonstrate that in such settings a static protocol realization of ZK is sufficient for ensuring adaptive security for the parent hybrid protocol something that enables simplified and substantially more efficient UC realizations of such protocols.

## 1 Introduction

When analyzing the security of cryptographic protocols there typically exists a divide between adaptive and static security, cf. [6]. In an adaptive security setting

---

\* Research partly supported by NSF CAREER Award CNS-0447808.

the adversary is allowed to corrupt parties dynamically and this makes simulation based proofs difficult: in particular without assuming erasures [2] the simulator would be forced to reconstruct the internal state of a corrupted machine that has been simulated. In fact, depending on the arguments used to prove the indistinguishability of simulated protocol transcripts, state reconstruction can be impossible. In contrast, in the static security setting, state reconstruction is not needed since the adversary is forced to decide a-priori on which parties are to be corrupted; this gives the leeway to the simulator to communicate to the adversary simulated transcripts that even though they substantially deviate from real protocol transcripts they are still indistinguishable from the point of view of a static adversary.

The divide between static and adaptive security in simulation based security proofs naturally impacts the complexity of attaining these levels of security for many cryptographic functionalities (both in terms of protocol efficiency as well as in terms of necessary idealized setup assumptions). In particular, for a given functionality, an adaptively secure protocol realizing it, is typically much more complicated compared to a protocol that only realizes it in the static sense. In the Universal Composition (UC) setting most interesting functionalities can be realized much more easily in the static security sense; (a notable exception is the ideal functionality of a digital signature [4,5]). This holds true also for the Zero-Knowledge ideal functionality  $\mathcal{F}_{\text{ZK}}$  that idealizes the operation of a zero-knowledge protocol [5]. Realizing  $\mathcal{F}_{\text{ZK}}$  in the UC-setting is based on the notion of UC-commitment [7]. Obtaining UC-commitments in the adaptive security sense is a rather arduous task [9,10].

The functionality  $\mathcal{F}_{\text{ZK}}$  is arguably one of the most useful sub-component functionalities in the design of complex cryptographic protocols (cf. [12,11]). The UC setting gives us the flexibility to focus on how to realize  $\mathcal{F}_{\text{ZK}}$  with some protocol  $\rho$  individually; then, given such realization, the universal composition theorem [3,5] enables us to focus on protocol design in the  $\mathcal{F}_{\text{ZK}}$ -hybrid world.

While the design of protocols within the  $\mathcal{F}_{\text{ZK}}$ -hybrid world is particularly attractive (given the power of the included ideal functionality that is supplied “for free” in the hybrid world) one cannot undervalue the substantial cost that will be incurred when  $\mathcal{F}_{\text{ZK}}$  will be substituted with some protocol  $\rho$  that realizes the ideal functionality in the adaptive security sense. This brings forth the following fundamental question that is the central theme of the present work: Are there useful  $\mathcal{F}_{\text{ZK}}$ -like functionalities that are (1) substantially cheaper to realize than  $\mathcal{F}_{\text{ZK}}$  against adaptive adversaries and (2) still sufficiently powerful to be useful as  $\mathcal{F}_{\text{ZK}}$  substitutes within a certain UC modular design scenario? Or, to pose this question more specifically, is it always necessary to use an adaptively secure realization of the ZK functionality in order to preserve the adaptive security of an  $\mathcal{F}_{\text{ZK}}$  hybrid protocol under composition?

**Contributions.** In this work we answer the question posed above. In particular we define the ideal functionality of “leaking zero-knowledge”  $\mathcal{F}_{\text{LZK}}$  that has the following characteristics:

- (1) The leaking zero-knowledge functionality  $\mathcal{F}_{\text{LZK}}$  is based on  $\mathcal{F}_{\text{ZK}}$  with the difference that it leaks to the adversary some information about the witness in a controlled way: in particular  $\mathcal{F}_{\text{LZK}}$  encompasses a specialized commitment scheme (that we call  $R$ -commitment where  $R$  is the ZK-relationship and we formalize herein) and when the prover issues a “prove” command to the functionality  $\mathcal{F}_{\text{LZK}}$ , the functionality leaks a commitment to  $w$  to the adversary. If the prover is corrupted at any moment after the commitment has been released, the commitment is opened to the adversary.
- (2) We prove that  $\mathcal{F}_{\text{LZK}}$  is interchangeable with  $\mathcal{F}_{\text{ZK}}$  against static adversaries. Thus in some sense, one can say, that  $\mathcal{F}_{\text{LZK}}$  is a “static version” of the  $\mathcal{F}_{\text{ZK}}$  zero-knowledge functionality. This also immediately implies that as long as one is interested in static security,  $\mathcal{F}_{\text{LZK}}$  can be used in place of  $\mathcal{F}_{\text{ZK}}$ . Moreover, it hints that  $\mathcal{F}_{\text{LZK}}$  may be “cheaper” to realize against adaptive adversaries when compared to  $\mathcal{F}_{\text{ZK}}$ . Indeed we present a simple protocol that realizes  $\mathcal{F}_{\text{LZK}}$  in the  $(\mathcal{F}_{\text{PRS}}, \mathcal{F}_{\text{ZKPM}})$ -hybrid world against adaptive adversaries (and thus automatically also  $\mathcal{F}_{\text{ZK}}$  against static adversaries); it seems difficult to obtain a protocol of similar complexity that realizes  $\mathcal{F}_{\text{ZK}}$  against adaptive adversaries in the  $(\mathcal{F}_{\text{PRS}}, \mathcal{F}_{\text{ZKPM}})$ -hybrid world.
- (3) It is possible to construct an environment that uses adaptive corruptions and separates the two functionalities  $\mathcal{F}_{\text{LZK}}$  and  $\mathcal{F}_{\text{ZK}}$ , unless the involved ZK-relation is a trivial relationship (to be clarified further in section 3.3). Moreover, we show that  $\mathcal{F}_{\text{ZK}}$  emulates  $\mathcal{F}_{\text{LZK}}$  against any adversary something that is indicative of the fact that  $\mathcal{F}_{\text{ZK}}$  is more powerful as a functionality.
- (4) In the adaptive adversary setting, we characterize a family of protocols (using a sufficient condition cf. section 5.2) that operate in the  $\mathcal{F}_{\text{ZK}}$ -hybrid world and have the property that they *retain adaptive security when transported to the  $\mathcal{F}_{\text{LZK}}$ -hybrid world*. To put it simply, for such protocols using  $\mathcal{F}_{\text{ZK}}$  is an “overkill” and it would be sufficient to consider them as protocols in the  $\mathcal{F}_{\text{LZK}}$ -hybrid world.

Interpreting the above in the context of the  $\mathcal{F}_{\text{ZK}}$ -hybrid world leads to the somewhat surprising result that there exist protocols where a certain static security realization of  $\mathcal{F}_{\text{ZK}}$  (which is an adaptive realization of  $\mathcal{F}_{\text{LZK}}$ ) is *still sufficient to achieve adaptive security* in the UC setting. In such settings we can say that we have traded static for adaptive security!

As expected the family of protocols we characterize in item (4) above excludes many functionalities that apparently require the adaptive security properties of a realization of  $\mathcal{F}_{\text{ZK}}$ . Still, many useful protocols fall into the class of protocols that we can trade static for adaptive security. In fact, the class, intuitively, contains all protocols that employ  $\mathcal{F}_{\text{ZK}}$  for “consistency purposes” (rather than say for witness hiding purposes).

A simple example of a protocol that belongs to the class is the usage of the  $\mathcal{F}_{\text{ZK}}$  functionality that is part of the adaptive commit-and-prove protocol ACP of [9]:

the ACP protocol involves three different instances of the  $\mathcal{F}_{\text{ZK}}$  functionality where one of them (the one employed by the verifier to ensure that his commitment key is valid) can in fact be substituted by  $\mathcal{F}_{\text{LZK}}$  without affecting the protocol’s adaptive security (cf.  $\mathcal{F}_{\text{ZK}}^{\text{T}}$  in Figure 10, page 57 in [9]; we note that the  $\mathcal{F}_{\text{ZK}}^{\text{T}}$  functionality can also be simulated by an  $\mathcal{F}_{\text{CRS}}$  box — a fact remarked in [9]). A more complex example of usage of  $\mathcal{F}_{\text{ZK}}$  within a protocol that can be substituted by  $\mathcal{F}_{\text{LZK}}$  is exhibited in [13] for the design of UC blind signatures: in this type of signatures it turns out that the signer requires only  $\mathcal{F}_{\text{LZK}}$  (as opposed to  $\mathcal{F}_{\text{ZK}}$  that is required for the user side).

**Other related work.** Relaxations of ideal functionalities were also seen in the context of the “monitored functionalities” of [15]; note that the goal there was to relax w.r.t. correctness rather than security as we do here. A relaxation w.r.t. security for the key-exchange ideal functionality was performed in [8]; in their setting the ideal-functionality leaks a function of the exchanged key (by including the so called “non-information” oracle).

**Notations.**  $a \xleftarrow{\text{RND}}$  denotes randomly selecting  $a$  in its domain;  $\text{negl}()$  denotes negligible function.

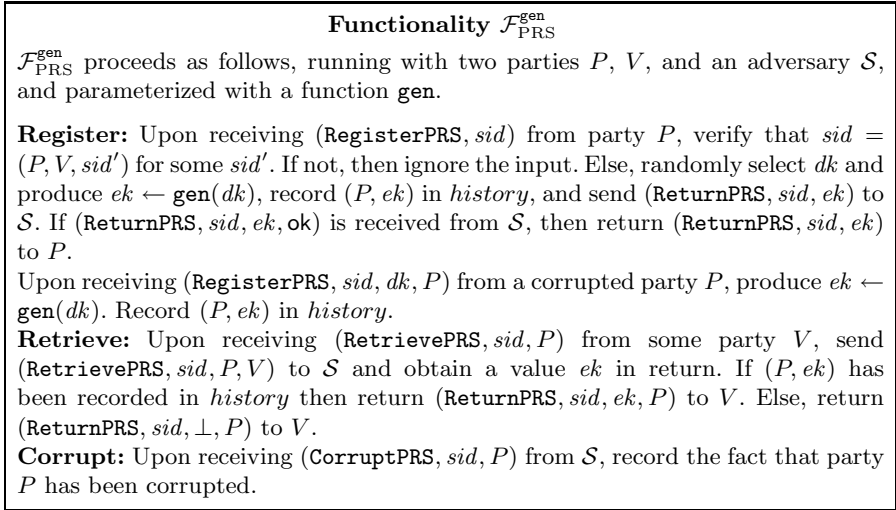
## 2 Preliminaries

**The Universal Composability Framework [5].** Defining security in the universal composability framework involves the following steps: we first specify an ideal functionality  $\mathcal{F}$ , which describes the desired behavior of the protocol by using a trusted party; this functionality  $\mathcal{F}$  communicates also with an ideal world adversary. Then, we prove that a particular protocol  $\pi$  operating in the real world securely realizes this ideal functionality. Here, securely realizing means that for any adversary  $\mathcal{A}$  in the real world, there exists a simulator  $\mathcal{S}$  in the ideal world, and no environment  $\mathcal{Z}$  can distinguish its interaction with the real protocol  $\pi$  and  $\mathcal{A}$ , or with the functionality  $\mathcal{F}$  and  $\mathcal{S}$ . Once this is established, we can take advantage of the UC composition theorem and “plug in” the protocol  $\pi$  as a sub-routine in any arbitrary environment in place of the functionality  $\mathcal{F}$ . For a complete definition of UC framework please refer to [5].

**Functionality  $\mathcal{F}_{\text{PRS}}^{\text{gen}}$ .** Next we describe functionality  $\mathcal{F}_{\text{PRS}}^{\text{gen}}$ , which is similar to the KS, KR, PRS functionalities employed respectively in [5, 11, 14]. Here we only consider the case for two parties,  $P$  and  $V$  (and thus we modify it accordingly).

**Functionality  $\mathcal{F}_{\text{ZK}}^{\text{R}}$ .** A zero-knowledge proof is a two-party protocol parameterized by a binary relation  $R$ ; the two parties called the prover and the verifier share a common input, the statement  $x$ . The prover has an additional input, the witness  $w$ . If  $(x, w) \in R$ , the verifier accepts; if not, the verifier will reject. Furthermore the verifier learn nothing from the protocol with the prover except of whether the prover knows the witness  $w$  s.t.  $(x, w) \in R$  or not. The functionality in figure 2 is taken from [5] which captures properly the security properties of a zero-knowledge proof.





**Fig. 1.** Private reference string functionality  $\mathcal{F}_{\text{PRS}}^{\text{gen}}$  for two parties

### 3 The Leaking Zero-Knowledge Functionality

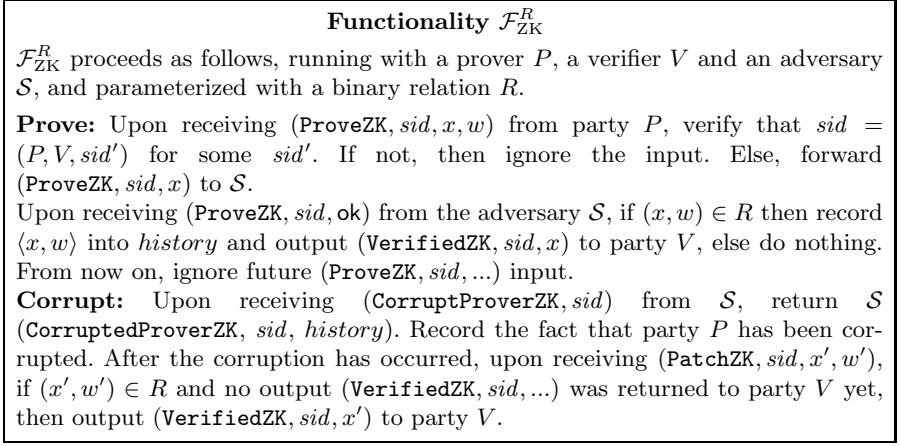
#### 3.1 R-Commitment

An  $R$ -commitment scheme is a special non-interactive commitment scheme that is bound to a given relation  $R$ . It is an extractable commitment where the hiding property is only required to hold with respect to the witnesses of the relation  $R$ . In particular, if the witness is computationally hidden by the statement, then an  $R$ -commitment can be at most computationally hiding. Formally, an  $R$ -commitment scheme  $\mathcal{E}$  is a tuple  $\langle \text{gen}_{\mathcal{E}}, \text{com}_{\mathcal{E}}, \text{ver}_{\mathcal{E}}, \text{dec}_{\mathcal{E}} \rangle$ . The key generation algorithm  $\text{gen}_{\mathcal{E}}$  produces a public parameter  $ek$  based on a randomly selected  $dk \in \mathcal{K}$ . The procedures  $\text{com}_{\mathcal{E}}, \text{ver}_{\mathcal{E}}$  correspond to the commitment algorithm and the testing algorithm for the decommit information for a given commitment; they satisfy the correctness property  $\text{ver}_{\mathcal{E}}(x, ek, \text{com}_{\mathcal{E}}(ek, x, w, \gamma), w, \gamma) = 1$  for any  $ek \leftarrow \text{gen}_{\mathcal{E}}(dk)$  with  $dk \in \mathcal{K}$ . The procedure  $\text{dec}_{\mathcal{E}}$  always extracts the witness given the trapdoor key  $dk$ ; in particular, we require  $\forall E \exists w, \gamma$  such that  $E = \text{com}_{\mathcal{E}}(ek, x, w, \gamma)$  and  $\text{dec}_{\mathcal{E}}(x, ek, E, dk) = w$ . Note that we may generalize these requirements to allow for partial correctness and extractability but this would not have any significant impact on our results.

We say that  $\mathcal{E}$  is an  $R$ -commitment for a given relation  $R$  if additionally to the above, it satisfies the ***R-hiding*** property:

**Definition 1 (R-hiding).** *We say a commitment  $\mathcal{E}$  is  $R$ -hiding, if for all PPT adversaries  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ , the advantage  $\text{Adv}_{\text{hiding}}^{R, \mathcal{E}}(\lambda) \stackrel{\text{def}}{=} |2\text{Prob}[\text{Exp}_{\text{hiding}}^{R, \mathcal{E}}(\lambda)] - 1| = \text{negl}(\lambda)$ , where the experiment  $\text{Exp}_{\text{hiding}}^{R, \mathcal{E}}(\lambda)$  is defined below.*

Additionally, we say that  $\mathcal{E}$  is ***R-unequivocal*** for some  $\text{sample}_R$  if it satisfies:



**Fig. 2.** Zero-knowledge functionality  $\mathcal{F}_{\text{ZK}}^R$

**Definition 2 (R-unequivocal).** We say a commitment  $\mathcal{E}$  is *R-unequivocal* for some PPT  $\text{sample}_R$  that returns  $(x, w)$  in  $R$ , if for all PPT adversaries  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ , the advantage  $\text{Adv}_{\text{unequivocal}}^{R, \mathcal{E}}(\lambda) \stackrel{\text{def}}{=} \text{Prob}[\text{Exp}_{\text{unequivocal}}^{R, \mathcal{E}}(\lambda) = 1] = \text{negl}(\lambda)$ , where the experiment  $\text{Exp}_{\text{unequivocal}}^{R, \mathcal{E}}(\lambda)$  is defined below.

|  |
|--|
| $\text{Exp}_{\text{unequivocal}}^{R, \mathcal{E}}(\lambda)$<br>$(x, w) \leftarrow \text{sample}_R(1^\lambda);$<br>$(ek, \hat{E}, \text{state}) \leftarrow \mathcal{A}_1(x);$<br>$\gamma \leftarrow \mathcal{A}_2(\text{state}, w);$<br>if $\text{ver}_{\mathcal{E}}(x, ek, \hat{E}, w, \gamma) = 1$<br>then output 1<br>else output 0. |
|--|

|   |
|---|
| $\text{Exp}_{\text{hiding}}^{R, \mathcal{E}}(\lambda)$<br>$(x, w) \leftarrow \mathcal{A}_1(1^\lambda);$ if $\text{verify}_R(x, w) \neq 1$ then abort;<br>$dk \xleftarrow{\mathcal{K}} \mathcal{K}; ek \leftarrow \text{gen}_{\mathcal{E}}(dk); b \xleftarrow{\mathcal{R}} \{0, 1\};$<br>if $b = 0$ then $E \leftarrow \text{com}_{\mathcal{E}}(x, ek, \hat{w}, \hat{\gamma}); \hat{w}, \hat{\gamma} \xleftarrow{\mathcal{R}} \text{RND};$<br>else $E \leftarrow \text{com}_{\mathcal{E}}(x, ek, w, \gamma); \gamma \xleftarrow{\mathcal{R}} \text{RND};$<br>$b^* \leftarrow \mathcal{A}_2(x, w, E);$<br>if $b^* = b$ then return 1 else return 0. |
|---|

### 3.2 Functionality $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$

In this subsection we introduce our new ZK functionality, called the leaking zero-knowledge functionality,  $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$ , in figure 3; it is parameterized by a relation  $R$  as well as an  $R$ -commitment  $\mathcal{E}$ . The design of  $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$  is based on  $\mathcal{F}_{\text{ZK}}^R$ . Recall that in the “prove” stage of  $\mathcal{F}_{\text{ZK}}^R$ , upon receiving the statement-witness pair  $\langle x, w \rangle$ ,  $\mathcal{F}_{\text{ZK}}^R$  is supposed to communicate the statement  $x$  to the adversary (but not the witness). In our case, during the “prove” stage of  $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$ , we allow  $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$  to leak more information about the witness that includes the parameter  $ek$  and a commitment  $E$  of the witness  $w$ , that is based on the parameter  $ek$ .

Note that we still anticipate  $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$  to capture some level of the zero-knowledge property, and a computationally bounded adversary still would not obtain any

useful information about the witness  $w$  from reading the extra information  $ek$  and  $E$  that is leaked together with the statement (this is based on the  $R$ -hiding property of the commitment as described above). Still, the “quality” of zero-knowledge offered by  $\mathcal{F}_{LZK}^{R,\mathcal{E}}$  is substantially impaired compared to  $\mathcal{F}_{ZK}^R$ . Note that whenever the prover is corrupted the commitment that was issued for proof’s witness will be opened (i.e., the adversary will not only receive the witness but also the decommitment information of the released commitment).

**Functionality  $\mathcal{F}_{LZK}^{R,\mathcal{E}}$**

$\mathcal{F}_{LZK}^{R,\mathcal{E}}$  proceeds as follows, running with a prover  $P$ , a verifier  $V$  and an adversary  $\mathcal{S}$ , and parameterized with a binary relation  $R$  and an  $R$ -commitment  $\mathcal{E}$ ; it incorporates  $\mathcal{F}_{PRS}^{\text{gen}\mathcal{E}}$  and furthermore it has the additional functions as below.

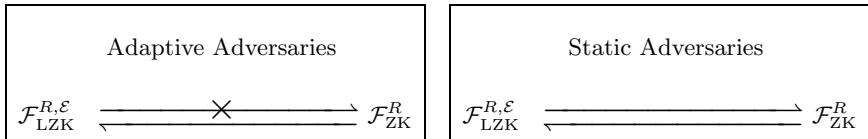
**Prove:** Upon receiving  $(\text{ProveLZK}, sid, x, w)$  from party  $P$ , verify that  $sid = (P, V, sid')$  for some  $sid'$ . If not, then ignore the input. Else, randomly select  $\gamma$  and compute  $E = \text{com}_{\mathcal{E}}(x, ek, w, \gamma)$ , and forward  $(\text{ProveLZK}, sid, x, ek, E)$  to  $\mathcal{S}$ . Upon receiving  $(\text{ProveLZK}, sid, \text{ok})$  from the adversary  $\mathcal{S}$ , if  $(x, w) \in R$  then record  $\langle x, E, w, \gamma \rangle$  in *history* and output  $(\text{VerifiedLZK}, sid, x)$  to party  $V$ , else do nothing. From now on, ignore future  $(\text{ProveLZK}, sid, \dots)$  input.

**Corrupt:** Upon receiving  $(\text{CorruptProverLZK}, sid)$  from  $\mathcal{S}$ , return  $\mathcal{S}$   $(\text{CorruptedProverLZK}, sid, \text{history}, ek)$ . Record this fact that party  $P$  has been corrupted. After the corruption has occurred, upon receiving  $(\text{PatchLZK}, sid, x', w')$ , if  $(x', w') \in R$  and no output  $(\text{VerifiedLZK}, sid, \dots)$  was returned to party  $V$  yet, then output  $(\text{VerifiedLZK}, sid, x')$  to party  $V$ .

**Fig. 3.** Leaking zero-knowledge functionality  $\mathcal{F}_{LZK}^{R,\mathcal{E}}$

### 3.3 Relation Between $\mathcal{F}_{LZK}^{R,\mathcal{E}}$ and $\mathcal{F}_{ZK}^R$

In this subsection we explore the essential relation between  $\mathcal{F}_{LZK}^{R,\mathcal{E}}$  and  $\mathcal{F}_{ZK}^R$ . First, we show that the functionality  $\mathcal{F}_{ZK}^R$  can UC-emulate  $\mathcal{F}_{LZK}^{R,\mathcal{E}}$ ; on the other hand, the other direction can only hold against static adversaries. Please refer to figure 4 below.



**Fig. 4.** Relation between  $\mathcal{F}_{LZK}^{R,\mathcal{E}}$  and  $\mathcal{F}_{ZK}^R$ .  $\mathcal{F}_1 \rightarrow \mathcal{F}_2$  stands for “ $\mathcal{F}_1$  UC-emulates  $\mathcal{F}_2$ .”

To establish the emulation result we show that a dummy protocol in the  $\mathcal{F}_{ZK}^R$ -hybrid world realizes  $\mathcal{F}_{LZK}^{R,\mathcal{E}}$ . It is easy to see that a simulator interacting with  $\mathcal{F}_{LZK}^{R,\mathcal{E}}$  can perfectly simulate transcripts to an environment that operates with dummy parties in the  $\mathcal{F}_{ZK}^R$ -hybrid world by simply suppressing the extra information provided by  $\mathcal{F}_{LZK}^{R,\mathcal{E}}$ .

**Theorem 1.** Let  $\mathcal{F}_{\text{ZK}}^R$  be the ideal ZK functionality, and  $\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}$  be the leaking version of  $\mathcal{F}_{\text{ZK}}^R$ . Let  $\rho_d$  be a dummy ZK protocol. Then for any adversary  $\mathcal{A}$  there exists an adversary  $\mathcal{S}$  such that for any adaptive environment machine  $\mathcal{Z}$  we have:  $\text{EXEC}_{\rho_d, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{ZK}}^R} = \text{EXEC}_{\rho_d, \mathcal{S}, \mathcal{Z}}^{\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}}$ .

We then investigate the other direction of theorem 1; we prove that a dummy protocol in  $\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}$ -hybrid world can *statically* realize functionality  $\mathcal{F}_{\text{ZK}}^R$  as described in theorem 2. The simulation is not perfect as it relies on the hiding properties of the  $R$ -commitment  $\mathcal{E}$ .

**Theorem 2.** Let  $\mathcal{F}_{\text{ZK}}^R$  be the ideal ZK functionality,  $\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}$  the leaking version of  $\mathcal{F}_{\text{ZK}}^R$ , and  $\rho_d$  a dummy ZK protocol. If  $\mathcal{E}$  is an  $R$ -hiding commitment, then for any adversary  $\mathcal{A}$  there exists an adversary  $\mathcal{S}$  such that for any static environment machine  $\mathcal{Z}$  we have:  $|\text{EXEC}_{\rho_d, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}} - \text{EXEC}_{\rho_d, \mathcal{S}, \mathcal{Z}}^{\mathcal{F}_{\text{ZK}}^R}| \leq \text{Adv}_{\text{hiding}}^{R,\mathcal{E}}(\lambda)$ .

Regarding adaptive adversaries, we *cannot* extend the result of the previous theorem. We establish this in theorem 3. The basic reason is that in the simulation of theorem 2 the simulator for  $\mathcal{F}_{\text{ZK}}^R$  has to simulate the extra information  $\langle ek, E \rangle$ . The simulator can easily simulate  $ek$  by just using the key-generator  $\text{gen}_{\mathcal{E}}$ . However the simulator gets in trouble when it needs to simulate  $E$  for an adaptive environment  $\mathcal{Z}$ . Note that the simulator does not know the witness  $w$ , which is “blocked” inside the functionality  $\mathcal{F}_{\text{ZK}}^R$ . The simulator may produce  $E$  based on a fake witness or simulate  $E$  in some other way; but when the adaptive  $\mathcal{Z}$  corrupts the prover after the simulated commitment has been released, the simulator must explain  $E$  to  $\mathcal{Z}$  for the real witness (that is now released from the ideal functionality). This would require that the underlying  $R$ -commitment scheme to be “equivocal” (which it is not).

Given the inflexibility of the  $R$ -commitment the simulator may still succeed if the relation  $R$  is somewhat trivial, and an adversary can obtain the correct witness by observing the statement  $x$ . In such case, the simulator now has chance to develop a successful simulation even if the environment is adaptive. The  $R$ -unequivocal property was designed appropriately so that it captures all these scenarios; based on this, we obtain the following theorem that demonstrates that the functionality  $\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}$  is weaker as a security notion compared to  $\mathcal{F}_{\text{ZK}}^R$ .

**Theorem 3.** Let  $\mathcal{F}_{\text{ZK}}^R$  be the ideal ZK functionality,  $\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}$  the leaking version of  $\mathcal{F}_{\text{ZK}}^R$ , and  $\rho_d$  a dummy ZK protocol. If  $\mathcal{E}$  is  $R$ -unequivocal for some  $\text{sample}_R$ , then there exists an adversary  $\mathcal{A}$  and an adaptive environment machine  $\mathcal{Z}$  such that for any adversary  $\mathcal{S}$ , we have:  $|\text{EXEC}_{\rho_d, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}} - \text{EXEC}_{\rho_d, \mathcal{S}, \mathcal{Z}}^{\mathcal{F}_{\text{ZK}}^R}| \geq 1 - \text{Adv}_{\text{unequivocal}}^{R,\mathcal{E}}(\lambda)$ .

### 4 Implementation of $\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}$ in the $(\mathcal{F}_{\text{PRS}}^{\text{gen}_{\mathcal{E}}}, \mathcal{F}_{\text{ZKPM}}^{R'})$ -Hybrid World

In this section we show that  $\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}$  can be realized with the protocol  $\pi_{\text{LZK}}$  in the  $(\mathcal{F}_{\text{PRS}}^{\text{gen}_{\mathcal{E}}}, \mathcal{F}_{\text{ZKPM}}^{R'})$ -hybrid world presented in figure 5. Note that based on Nielsen’s

result (refer to theorem 5.1 in page 180 in Nielsen’s PhD thesis [14]),  $\mathcal{F}_{\text{ZKPM}}$  can be very efficiently implemented in the  $\mathcal{F}_{\text{PRS}}$ -hybrid world as  $\mathcal{F}_{\text{ZKPM}}$  does not require witness extraction. So  $\pi_{\text{LZK}}$  can be implemented in the  $\mathcal{F}_{\text{PRS}}$ -hybrid world without requiring UC commitments. Next we prove that the protocol  $\pi_{\text{LZK}}$  from figure 5 realizes  $\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}$ .

**Protocol  $\pi_{\text{LZK}}$  in the  $(\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}, \mathcal{F}_{\text{ZKPM}}^{R'})$ -Hybrid World**

On input (**ProveLZK**,  $sid, x, w$ ) from  $\mathcal{Z}$ , party  $P$  sends (**RegisterPRS**,  $sid$ ) to  $\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}$ . Whenever Party  $P$  receives (**ReturnPRS**,  $sid, ek$ ) from  $\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}$ , it randomly selects  $\gamma$  and computes  $E = \text{com}_{\mathcal{E}}(x, ek, w, \gamma)$ , and sends  $\mathcal{F}_{\text{ZKPM}}^{R'}$  the message (**ProveZKPM**,  $sid, (x, ek, E), (w, \gamma)$ ).

Whenever Party  $V$  receives (**VerifiedZKPM**,  $sid, (x, ek, E)$ ) from  $\mathcal{F}_{\text{ZKPM}}^{R'}$ , it sends (**GetPRS**,  $sid$ ) to  $\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}$  and get  $ek$  from  $\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}$ ; if the  $ek$  is same as the one from  $\mathcal{F}_{\text{ZKPM}}^{R'}$  then returns (**VerifiedLZK**,  $sid, x$ ) to  $\mathcal{Z}$ .

**Fig. 5.** A protocol realizing  $\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}$  in the  $(\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}, \mathcal{F}_{\text{ZKPM}}^{R'})$ -hybrid world. Here  $\mathcal{E}$  is an  $R$ -commitment, and relation  $R'$  is based on relation  $R$  and key generator  $\text{gen}$ , i.e.  $R' = \{(x, ek, E), (w, \gamma) \mid (x, w) \in R \wedge E = \text{com}_{\mathcal{E}}(x, ek, w, \gamma)\}$ .

**Theorem 4.** Consider protocol  $\pi_{\text{LZK}}$  in the  $(\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}, \mathcal{F}_{\text{ZKPM}}^{R'})$ -hybrid world in figure 5, where  $\mathcal{E}$  is an  $R$ -commitment. Let  $\pi_{\text{d}}$  be a dummy ZK protocol. Then for any adversary  $\mathcal{A}$  there exists an adversary  $\mathcal{S}$  such that for any adaptive environment machine  $\mathcal{Z}$  we have:  $\text{EXEC}_{\pi_{\text{LZK}}, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}, \mathcal{F}_{\text{ZKPM}}^{R'}} = \text{EXEC}_{\pi_{\text{d}}, \mathcal{S}, \mathcal{Z}}^{\mathcal{F}_{\text{LZK}}^{R,\mathcal{E}}}$ .

Based on theorem 2 and theorem 4, we obtain immediately that the protocol of figure 5 statically realizes  $\mathcal{F}_{\text{ZK}}^R$  (with an  $\text{Adv}_{\text{hiding}}^{R,\mathcal{E}}(\lambda)$  distance).

In general, we can design a protocol  $\pi_{\text{ZK}}$  to realize  $\mathcal{F}_{\text{ZK}}$  in the  $(\mathcal{F}_{\text{COM}}, \mathcal{F}_{\text{ZKPM}})$ -hybrid world by committing the witness and then using ZKPM to bind the commitment and the ZK statement as in the figure 5 where  $E$  is computed based on  $\mathcal{F}_{\text{COM}}$ . Note that in  $\pi_{\text{LZK}}$  we compute  $E$  based on the  $R$ -commitment, but in  $\pi_{\text{ZK}}$  we need the commitment to be both extractable and equivocal: in the case that the prover is corrupted,  $\mathcal{F}_{\text{ZK}}$  only supplies the witness and the simulator needs to figure out the random coins involved; on the contrary  $\mathcal{F}_{\text{LZK}}$  supplies all witness and coins for  $E$ . Combining equivocality and extractability seems that it requires more work (more rounds or more communication), cf. [9,10].

## 5 Using $\mathcal{F}_{\text{LZK}}$ in Place of $\mathcal{F}_{\text{ZK}}$

### 5.1 A Protocol Transformation

We describe a useful transformation which allows a protocol  $\pi$  in the  $\mathcal{F}_{\text{ZK}}^R$ -hybrid world to be modified into a slightly different protocol  $\tilde{\pi}$  based on an  $R$ -commitment  $\mathcal{E}$ . The protocol  $\tilde{\pi}$  operates in the  $(\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}, \mathcal{F}_{\text{ZK}}^{R'})$ -hybrid world for

a relation  $R'$  defined as follows:  $R' = \{(x, ek, E), (w, \gamma) \mid (x, w) \in R \wedge E = \text{com}_\mathcal{E}(x, ek, w, \gamma)\}$ , where  $ek$  is obtained from  $\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}$ ,  $\gamma$  is randomly selected. In section 5.2 we will use such transformation to explore the application of the functionality  $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$ .

**Transformation from  $\pi$  into  $\tilde{\pi}$**

Each time in protocol  $\pi$ , party  $P_\pi$  sends (**ProveZK**,  $sid, x, w$ ) to  $\mathcal{F}_{\text{ZK}}^R$ , in protocol  $\tilde{\pi}$  party  $P_{\tilde{\pi}}$  sends (**RegisterPRS**,  $sid$ ) to  $\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}$ ; when it receives (**ReturnPRS**,  $sid, ek$ ) from  $\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}$ , party  $P_{\tilde{\pi}}$  randomly selects  $\gamma$ , it computes  $E = \text{com}_\mathcal{E}(x, ek, w, \gamma)$  and sends (**ProveZK**,  $sid, (x, ek, E), (w, \gamma)$ ) to  $\mathcal{F}_{\text{ZK}}^{R'}$ .

Each time in protocol  $\tilde{\pi}$ , party  $V_{\tilde{\pi}}$  receives (**VerifiedZK**,  $sid, (x, ek, E)$ ) from  $\mathcal{F}_{\text{ZK}}^{R'}$ , it sends (**RetrievePRS**,  $sid, P_\pi$ ) to  $\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}$ , and obtains  $ek$  from  $\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}$ ; if  $ek$  is same as the one from  $\mathcal{F}_{\text{ZK}}^{R'}$ , then party  $V_{\tilde{\pi}}$  sends (**Verified**,  $sid, x$ ) to  $\mathcal{Z}$ .

**Fig. 6.** A transformation from  $\pi$  in the  $\mathcal{F}_{\text{ZK}}^R$ -hybrid world into  $\tilde{\pi}$  in the  $(\mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}}, \mathcal{F}_{\text{ZK}}^{R'})$ -hybrid world

Note that functionally the protocols  $\pi$  and  $\tilde{\pi}$  are identical; nevertheless, the protocol  $\tilde{\pi}$  is possibly exposing some more information to the adversary as compared to  $\pi$  with respect to the witnesses that are employed within the  $\mathcal{F}_{\text{ZK}}$  version. If an adversary can see little difference between  $\tilde{\pi}$  and  $\pi$  then we can use  $\mathcal{F}_{\text{LZK}}$  in place of  $\mathcal{F}_{\text{ZK}}$ . We elaborate on this in the next subsection.

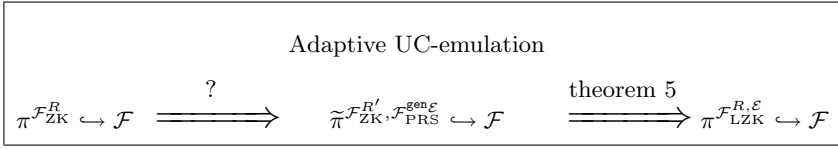
### 5.2 A Sufficient Condition

The goal of this section is to characterize the protocols for which we can substitute an  $\mathcal{F}_{\text{ZK}}^R$  implementation with a (potentially cheaper)  $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$  implementation in the setting of adaptive adversaries. The protocol transformation of the previous subsection serves as a “bridge” between the protocol in the  $\mathcal{F}_{\text{ZK}}$ -hybrid world and the protocol in the  $\mathcal{F}_{\text{LZK}}$ -hybrid world. We show in theorem 5 that if  $\pi$  realizes some  $\mathcal{F}$  and the transformed protocol  $\tilde{\pi}$  maintains this functionality, this implies that the original protocol  $\pi$  can be transported into the  $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$ -hybrid world without any impact.

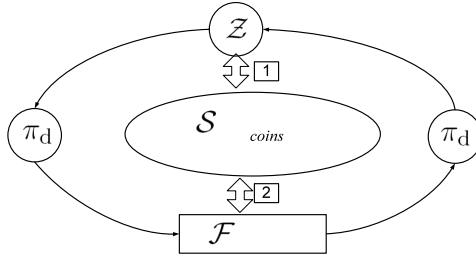
**Theorem 5.** (*Sufficient Condition*) *Let  $\pi$  be a protocol in the  $\mathcal{F}_{\text{ZK}}^R$ -hybrid-world,  $\tilde{\pi}$  the transformation of  $\pi$  as described in section 5.1. If  $\tilde{\pi}$  in the  $(\mathcal{F}_{\text{ZK}}^{R'}, \mathcal{F}_{\text{PRS}}^{\text{gen}\mathcal{E}})$ -hybrid world realizes functionality  $\mathcal{F}$ , then  $\pi$  in the  $\mathcal{F}_{\text{LZK}}^{R, \mathcal{E}}$ -hybrid world also realizes  $\mathcal{F}$ .*

The theorem is illustrated in figure 7. The sufficient condition for transporting a protocol  $\pi$  from the  $\mathcal{F}_{\text{ZK}}^R$  hybrid world into the  $\mathcal{F}_{\text{LZK}}$ -hybrid world is marked with “?”.

In the remaining of the section we investigate the setting where the sufficient condition can be satisfied. Assume a protocol  $\pi$  in the  $\mathcal{F}_{\text{ZK}}^R$ -hybrid-world that



**Fig. 7.** Trading  $\mathcal{F}_{LZK}$  for  $\mathcal{F}_{ZK}$ . Note that  $\pi^{\mathcal{F}_1} \hookrightarrow \mathcal{F}_2$  stands for “ $\pi$  realizes  $\mathcal{F}_2$  in the  $\mathcal{F}_1$ -hybrid world.”



**Fig. 8.** In constructing  $\tilde{S}$  the witness used for  $\mathcal{F}_{ZK}^R$  would be necessary for a simulation against adaptive adversaries;  $\tilde{S}$  may recover such witness if it appears in the communication lines 1, 2 or can be inferred from the coins of  $S$ .

realizes  $\mathcal{F}$ . This means there exists a simulator  $\mathcal{S}$  that can simulate  $\pi$ -protocol-transcripts for any adaptive  $Z$ . In particular  $\mathcal{S}$  simulates  $\mathcal{F}_{ZK}^R$  to produce the statement  $x$  and also the direct transcripts between the  $\pi$  parties.

In order to show that the transformed protocol  $\tilde{\pi}$  in the  $(\mathcal{F}_{ZK}^R, \mathcal{F}_{PRS}^{\text{gen}\varepsilon})$ -hybrid world also realizes  $\mathcal{F}$ , we need to construct a simulator  $\tilde{S}$  for the adaptive environment. We may build  $\tilde{S}$  based on  $\mathcal{S}$  which is given above by the assumption that  $\pi^{\mathcal{F}_{ZK}^R}$  realizes  $\mathcal{F}$ ; the statement  $x$  and the direct transcripts between the  $\pi$  parties can be simulated verbatim from  $\mathcal{S}$ . Still  $\tilde{S}$  needs to simulate the extra  $\langle ek, E \rangle$  information since  $\mathcal{S}$  does not supply this. Recall that our environment  $Z$  may involve adaptive corruptions. So  $\tilde{S}$  may not be able to produce the extra  $\langle ek, E \rangle$  based on a “fake” witness  $\tilde{w}$  (because when the prover is corrupted and a real witness  $w$  is supplied,  $\tilde{S}$  cannot explain  $E$  to the real  $w$ , cf. theorem 3). Excluding the case of a non  $R$ -unequivocal commitment (which is rather trivial), it follows that the only way for the proof to work would be if there are circumstances for which  $\tilde{S}$  is capable of inferring the witness from either the coins used by  $\mathcal{S}$  and or the “communication lines” of  $\mathcal{S}$  with  $Z$  or  $F$  as shown in 1 or/and 2 in figure 8. For example consider  $\mathcal{F}$  to be a functionality extending  $\mathcal{F}_{\text{SIG}}$  45 where the signer wishes to prove knowledge and correct application of his secret key to other parties (e.g., his signing key is involved in some more complex computation for meeting a certain goal of  $\mathcal{F}$ ). This is the case for example for the signer side in the UC blind signatures of 13; in this protocol, the signer would require only  $\mathcal{F}_{LZK}$  (as opposed to  $\mathcal{F}_{ZK}$ ) as the key is known to the simulator.

**Acknowledgements.** We thank Jesper Nielsen for his kind clarifications on [14]. We also thank the anonymous referees for their constructive comments.

## References

1. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS 2004, pp. 186–195 (2004)
2. Beaver, D., Haber, S.: Cryptographic protocols provably secure against dynamic adversaries. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 307–323. Springer, Heidelberg (1993)
3. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145 (2001)
4. Canetti, R.: Universally composable signature, certification, and authentication. In: CSFW 2004, pp. 219–235 (2004), <http://eprint.iacr.org/2003/239/>
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Cryptology ePrint Archive: Report 2000/067 (December 2005), Latest version at <http://eprint.iacr.org/2000/067/>
6. Canetti, R., Damgård, I., Dziembowski, S., Ishai, Y., Malkin, T.: On adaptive vs. non-adaptive security of multiparty protocols. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 262–279. Springer, Heidelberg (2001)
7. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
8. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC 2002, pp. 494–503, <http://www.cs.biu.ac.il/~lindell/PAPERS/uc-comp.ps>
10. Damgård, I., Nielsen, J.B.: Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002), <http://www.brics.dk/RS/01/41/BRICS-RS-01-41.pdf>
11. Goldreich, O.: Foundations of Cryptography- Basic Tools. Cambridge University Press, Cambridge (2001)
12. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC 1987, pp. 218–229 (1987)
13. Kiayias, A., Zhou, H.-S.: Equivocal blind signatures and adaptive UC-security. In: Cryptology ePrint Archive: Report 2007/132 (2007)
14. Nielsen, J.B.: On protocol security in the cryptographic model. Dissertation Series DS-03-8, BRICS (2003), <http://www.brics.dk/DS/03/8/BRICS-DS-03-8.pdf>
15. Prabhakaran, M., Sahai, A.: Relaxing environmental security: Monitored functionalities and client-server computation. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 104–127. Springer, Heidelberg (2005)



# A Characterization of Non-interactive Instance-Dependent Commitment-Schemes (NIC)

Bruce Kapron, Lior Malka, and Venkatesh Srinivasan

Department of Computer Science  
University of Victoria, BC, Canada  
bmkapron, liorma, venkat@cs.uvic.ca

**Abstract.** We provide a new characterization of certain zero-knowledge protocols as *non-interactive instance-dependent commitment-schemes* (NIC). To obtain this result we consider the notion of  $V$ -bit protocols, which are very common, and found many applications in zero-knowledge. Our characterization result states that a protocol has a  $V$ -bit zero-knowledge protocol if and only if it has a NIC. The NIC inherits its hiding property from the zero-knowledge property of the protocol, and vice versa.

Our characterization result yields a framework that strengthens and simplifies many zero-knowledge protocols in various settings. For example, applying this framework to the result of Micciancio et al. [18] (who showed that some problems, including GRAPH-NONISOMORPHISM and QUADRATIC-RESIDUOUSITY, *unconditionally* have a concurrent zero-knowledge proof) we easily get that arbitrary, monotone boolean formulae over a large class of problems (which contains, e.g., the complement of any random self-reducible problem) *unconditionally* have a concurrent zero-knowledge proof.

**Keywords:** zero-knowledge, commitment-schemes, random self-reducibility.

## 1 Introduction

Zero-knowledge protocols are two party protocols that enable one party (the prover) to convince another party (the verifier) of an assertion, with the guarantee that the verifier learns nothing but the truth of the assertion [14]. These protocols play a central role in the theory of cryptography, and they are also interesting from a complexity theoretic perspective because they facilitate the study of  $\mathbf{NP}$  through interaction and randomness.

Zero-knowledge protocols and cryptography heavily rely on commitment-schemes. For example, every language in  $\mathbf{NP}$  has a computational zero-knowledge (CZK) protocol [13,5] if bit commitment-schemes (equivalently, one-way functions [15,20]) exist. Consequently, many results about zero-knowledge protocols, and cryptography in general, are based on unproven assumptions.

Recently, Vadhan [27] gave a characterization of CZK, called the SZK/OWF-CHARACTERIZATION, which leads to the construction of a special scheme from any zero-knowledge protocol. Utilizing this scheme and the techniques already known from the conditional study of zero-knowledge, Vadhan was able to prove many results about CZK without relying on any unproven assumptions. A similar approach was applied by Nguyen and Vadhan [21] in the context of zero-knowledge proofs with efficient

provers<sup>1</sup>, and by Ong and Vadhan [22] in the context of zero-knowledge arguments. The works of [27][21][22] demonstrate that we can prove unconditional results about zero-knowledge protocols. This can be done by characterizing zero-knowledge protocols as special bit commitment-schemes, and then using these special schemes instead of bit commitment-schemes<sup>2</sup>.

We continue this line of research. That is, we construct special schemes from a specific class of zero-knowledge protocols, and then we use the special schemes instead of bit commitment-schemes. Our schemes are simply functions. That is, by restricting ourselves to a specific class of zero-knowledge protocols we are able to construct very simple non-interactive schemes. In contrast, the schemes of Vadhan [27] can be constructed from *any* zero-knowledge protocol, but they are interactive, and have an involved definition (similar in flavor to that of zero-knowledge protocols). We stress that although our schemes are constructed from specific zero-knowledge protocols, they can be used in other zero-knowledge protocols, and in various settings. That is, our characterization result yields a framework with wide applicability.

**Our Results.** We provide a new characterization of certain zero-knowledge protocols as special bit commitment-schemes. To obtain this result we consider the notion of V-bit protocols. Informally, in such protocols the prover sends the first message  $m_1$ , the verifier sends back a random bit  $b$ , the prover replies with a message  $m_2$ , and the verifier accepts or rejects. These protocols are very common in zero-knowledge. Examples include the perfect zero-knowledge (PZK) proof of [4] for GRAPH-ISOMORPHISM, the statistical zero-knowledge (SZK) proof of [19] for certain lattice problems, the SZK and PZK proofs of [24] for variants of STATISTICAL-DISTANCE (SD), and more.

We construct an efficient function  $f(x, b; r)$  from any V-bit zero-knowledge protocol for a promise-problem  $\Pi \stackrel{\text{def}}{=} \langle \Pi_Y, \Pi_N \rangle$ . The inputs to  $f$  are a string  $x$ , a bit  $b$ , and randomness  $r$ . The output  $y$  of  $f$  hides  $b$  when  $x$  is a YES instance, and binds to  $b$  when  $x$  is a NO instance. More precisely, given  $y = f(x, b; r)$ , if  $x \in \Pi_Y$ , then  $b$  cannot be determined from  $y$ , and if  $x \in \Pi_N$ , then  $y$  can be a commitment to either 0 or 1, but not both (i.e.,  $y \neq f(x, 1-b; r')$  for all  $r'$ ). Notice that unlike bit commitment-schemes, the hiding and the binding properties of  $f$  may not hold simultaneously. Since  $f$  is a non-interactive commitment-scheme for  $\Pi$ , we call  $f$  a *non-interactive instance-dependent commitment-scheme* (NIC). Using the techniques of [10][16] we get the following:

**Main result (informal).** A problem  $\Pi$  has a V-bit zero-knowledge protocol if and only if  $\Pi$  has a NIC.

The NIC  $f$  inherits its hiding property from the zero-knowledge property of the V-bit protocol, and vice versa. For example, the SZK protocols for the lattice problems of Micciancio and Vadhan [19] yield a statistically hiding NIC for these problems, and vice versa.

The notion of V-bit protocols is related to Cramer's notion of  $\Sigma$ -protocols [7]. These protocols are similar to V-bit protocols in that they are also 3-round public-coin

<sup>1</sup> A prover is *efficient* if given witness for input  $x$  it runs in time polynomial in  $|x|$ .

<sup>2</sup> The idea of replacing a bit commitment-scheme with a special scheme is due to Itoh et al. [16]. However, [16] construct a special scheme (different from that of [27][21]) for specific languages, whereas [27][21] provide a characterization result.

protocols, but instead of sending a bit  $b$ , the verifier sends a string  $e$ . However, if we consider  $V$ -bit *zero-knowledge* protocols, then the two notions are equivalent (the idea is to let  $e$  be the bit  $b$ , followed by zeroes [11]). Thus, our characterization result applies to  $\Sigma$ -protocols as well.

An immediate corollary to the characterization result is a transformation from  $V$ -bit *honest-verifier* zero-knowledge protocols to *dishonest-verifier*  $V$ -bit zero-knowledge protocols with *efficient provers*. The transformation preserves the zero-knowledge property of the original protocol. When we apply it to, e.g., the protocol of [24] for variants of SD we immediately get a zero-knowledge protocol with an efficient prover for these variants, a result previously proved in [19] using similar ideas.

To show that our characterization result yields a useful framework we prove that NIC can be combined in a monotone boolean formula fashion (i.e., with AND and OR connectors). For example, if  $f$  is a NIC for GRAPH-ISOMORPHISM, and  $g$  is a NIC for the lattice problems of [19], then our lemma states that, e.g.,  $f \wedge g$  and  $f \vee g$  are also NIC for the corresponding problems.

**Second result (informal).** The class of problems possessing NIC is closed under arbitrary monotone boolean formulae.

In addition, we prove that any random self-reducible (RSR) problem [2] has a perfectly hiding NIC. This folklore lemma follows from [26,25], but here we provide the proof for completeness. Let us see how combining these lemmas with our characterization result yields a very useful framework.

**Removing computational assumptions.** Our framework allows replacing the bit commitment-scheme in the protocol of Barak [3] with a NIC. The protocol inherits its zero-knowledge property from the hiding property of the NIC. For example, we get that if a problem has a perfectly hiding NIC, then it has a public-coin, round-efficient protocol (i.e., constant-round, with a negligible soundness error, and perfect completeness). The protocol is a **PZK** argument with a strict, polynomial-time *non-black-box* simulator. Notice that our protocol applies to problems that have a NIC, whereas the protocol of [3] applies to all of **NP**. As in [3], our protocol assumes the existence of collision-resistant hash functions. However, our result yields **PZK** protocols (as opposed to **CZK** in [3]), and it does not use bit commitment-schemes.

**Abstraction and closure.** Our framework strengthens and simplifies the result of Micciancio, Ong, Sahai, and Vadhan [18], who showed that a NIC with reversed properties<sup>3</sup> can replace the bit commitment-scheme in the protocol of [23]. Unlike [18], since we already have a characterization result, we do not need to construct such a NIC for specific problems (e.g., GRAPH-NONISOMORPHISM) or to be familiar with their definition (e.g., the lattice problems of [19]). Also, our framework shows that such NIC are closed under monotone boolean formulae. Thus, when we apply our framework to the theorem of [18] we get that arbitrary, monotone boolean formulae over a large class of problems (which contains, e.g., the complement of any random self-reducible problem)

<sup>3</sup> By "reversed" we mean that the hiding property holds on NO instances of the problem (instead of YES instances), and the binding property holds on YES instances (instead of NO instances).

*unconditionally* have a concurrent zero-knowledge proof. Similar improvements apply to local zero-knowledge [17], and quantum zero-knowledge [28].

**Unifying previous works.** Our framework unifies under the theme of NIC the results of Tompa and Woll [26], De Santis, Di Crescenzo, Persiano, and Yung [25], and Itoh, Ohta, and Shizuya [16]. Actually, these works only consider the perfect setting, and focus mainly on RSR problems. In contrast, our framework includes problems that are not known to be RSR, and it also considers the statistical and the computational setting. Hence, we get stronger and more general results under one simple theme.

**Related work.** We use the idea of Damgård [10] to obtain a NIC from any  $V$ -bit zero-knowledge protocol. Feige and Shamir used a similar idea to construct a trapdoor commitment-scheme from a bit commitment-scheme. Notice that the context of the work of Damgård [10] was to investigate whether zero-knowledge imply bit commitment-schemes. That is, [10] constructed an interactive bit commitment-scheme (as opposed to a non-interactive, *instance-dependent* commitment-scheme) from a proof of knowledge for any **NP**-hard relation, provided that the proof is a  $\Sigma$ -protocol. In contrast, we construct a NIC from any  $V$ -bit zero-knowledge protocol, regardless of whether the underlying problem is **NP**-hard. Also, the binding property of our NIC follows from the soundness of the underlying  $V$ -bit protocol, whereas in [10] the binding property is computational, and follows from the hardness of the underlying problem.

Our lemma on the closure of NIC under monotone boolean formulae uses the ideas of [25]. These ideas were also used in [24,27] to show closure properties. Our lemma is related to the closure results of Damgård and Cramer [9], and Cramer, Damgård, and Mackenzie [8]. All these results are proved by modifying the original protocols to obtain the closure. In contrast, we prove our closure result in a simple combinatorial setting (using NIC), and we always use the same underlying protocol of Blum [5] for **NP**. In addition, the results of [9,8] change the properties of the original protocol. For example, in [9] the protocol becomes a private-coin protocol, and in [8] the protocol becomes a 4-round protocol. In contrast, since we work with NIC, our underlying protocol does not change.

Our NIC is related to versions of SD, a complete problem for **SZK** [24]. That is, a problem has a perfectly (respectively, statistically) hiding NIC if and only if it Karp-reduces to  $\overline{\text{SD}}^{1,0}$  (respectively,  $\overline{\text{SD}}^{1,1/2}$ ). The notion of a perfectly hiding NIC is implicit in [4], and formalized in [16]. The notion of a statistically hiding NIC was formalized by [19]. Here we provide the computational analogue.

## 2 Non-interactive, Instance-Dependent Commitment-Schemes

We define non-interactive, instance-dependent commitment-schemes (NIC). Using the technique of [16] we show that if a problem has a NIC, then it has a  $V$ -bit zero-knowledge protocol (this holds for computationally hiding NIC if, in addition, the problem is in **NP**). The protocol is also a proof of knowledge, and it inherits its zero-knowledge property from the hiding property of the NIC.

Intuitively, a *bit commitment-scheme* allows a sender to commit to a bit  $b$  such that the receiver cannot learn the value of  $b$ , yet the sender cannot change  $b$ . Informally, a NIC

is a bit commitment-scheme in which the hiding and the binding properties depend on a string  $x$ , and thus may not hold simultaneously. That is, instead of  $f(b; r)$  we consider  $f(x, b; r)$ , and the hiding and binding properties depend on whether  $x$  is a YES on a NO instance of some problem  $\Pi$ . Formally,

**Definition 2.1** (NIC). *Let  $\Pi = \langle \Pi_Y, \Pi_N \rangle$  be a promise-problem, and let  $f(x, b; r)$  be a probabilistic, polynomial-time Turing machine on inputs  $x$  and  $b \in \{0, 1\}$ . The string  $r$  denotes the randomness of  $f$ .*

*We say that  $f$  is **binding** on  $\Pi_N$  if for any  $x \in \Pi_N$ , and for any  $r$  and  $r'$  it holds that  $f(x, 0; r) \neq f(x, 1; r')$ . We say that  $f$  is perfectly (respectively, statistically, computationally) **hiding** on  $\Pi_Y$  if for any  $x \in \Pi_Y$  and each  $b \in \{0, 1\}$  the ensembles  $\{f(x, 0)\}_{x \in \Pi_Y}$  and  $\{f(x, 1)\}_{x \in \Pi_Y}$  are statistically identical (respectively, statistically indistinguishable, computationally indistinguishable).*

*We say that  $f$  is a perfectly (respectively, statistically, computationally) hiding NIC for  $\Pi$  if  $f$  is binding on  $\Pi_N$ , and perfectly (respectively, statistically, computationally) hiding on  $\Pi_Y$ .*

When appropriate we will omit the random input  $r$  to  $f$ . Notice that if  $f$  is a perfectly or a statistically hiding NIC for  $\Pi$ , then as a class of problems  $\mathbf{NP}$  contains  $\Pi$ . This is so because if  $x \in \Pi_Y$ , then there is a pair  $\langle r, r' \rangle$  such that  $f(x, 0; r) = f(x, 1; r')$ , and if  $x \in \Pi_N$ , then no such pair exists. However,  $\Pi$  may not be in  $\mathbf{NP}$  if  $f$  is computationally hiding. We give an example of a perfectly hiding NIC.

*Example 2.1.* NIC for the language GRAPH-ISOMORPHISM [4,16]. Let  $f(x, b; r)$  be a function that given a pair of graphs  $x = \langle G_0, G_1 \rangle$  on  $n$  vertices uses  $r$  to define a random permutation  $\pi$  over  $\{1, \dots, n\}$ , and outputs  $y = \pi(G_b)$ . If the graphs are isomorphic, then  $y$  is isomorphic to both  $G_0$  and  $G_1$ , and  $b$  cannot be determined from  $y$ . Conversely, if the graphs are not isomorphic, then  $y$  cannot be isomorphic to both  $G_0$  and  $G_1$ . Thus,  $f$  is a perfectly hiding NIC for GRAPH-ISOMORPHISM.

Our protocol follows the idea of [16], which uses the protocol of Blum [5] for the  $\mathbf{NP}$ -complete problem HAMILTONIAN-CIRCUIT (HC). In the protocol of [16] the prover and the verifier initially reduce the input  $x$  of the problem possessing a NIC to an instance  $G$  of HC, and then execute the zero-knowledge protocol of [5] using the NIC as a bit commitment-scheme. Notice that the prover can transform its witness for  $x$  into a witness for  $G$ , and thus it is efficient. When  $x \in \Pi_Y$  the scheme is hiding, and thus the protocol is zero-knowledge. When  $x \in \Pi_N$  the scheme is binding, and thus the protocol is sound. Our lemma follows. The proof is very similar to that of [16].

**Lemma 2.1.** *If a problem  $\Pi$  has a perfectly (respectively, statistically) hiding NIC, then  $\Pi$  has a public-coin **PZK** (respectively, **SZK**) proof with an efficient prover. If  $\Pi \in \mathbf{NP}$ , and  $\Pi$  has a computationally hiding NIC, then  $\Pi$  has a public-coin **CZK** proof with an efficient prover.*

Itoh, Ohta, and Shizuya [16] observed that if  $\Pi$  has a statistically hiding NIC, then  $\Pi$  cannot be  $\mathbf{NP}$ -complete, unless the polynomial hierarchy collapses [12,16]. In the next section we show that  $V$ -bit zero-knowledge protocols and NIC are equivalent. Thus,  $\mathbf{NP}$ -complete languages cannot have  $V$ -bit **SZK** proofs, unless the polynomial hierarchy collapses.

### 3 Characterizing V-Bit Zero-Knowledge Protocols

We introduce the notion of V-bit protocols, and then show how to construct a NIC from a simulator of any V-bit zero-knowledge protocol. Since the zero-knowledge protocols constructed in Section 2 for problems possessing NIC are V-bit zero-knowledge protocols, we get our main theorem.

**Theorem 3.1.** *A promise-problem  $\Pi$  has a V-bit **PZK** (respectively, **SZK**) proof if and only if  $\Pi$  has a perfectly (respectively, statistically) hiding NIC. Similarly,  $\Pi$  has a V-bit **CZK** proof if and only if  $\Pi \in \mathbf{NP}$  and  $\Pi$  has a computationally hiding NIC.*

We present the definition of V-bit protocols.

**Definition 3.1 (V-bit protocol).** *Let  $\Pi = \langle \Pi_Y, \Pi_N \rangle$  be a problem, and let  $\langle P, V \rangle$  be a protocol for  $\Pi$  with perfect completeness. We say that  $\langle P, V \rangle$  is V-bit if for any  $x \in \Pi_Y$  the interaction between  $P$  and  $V$  is as follows:  $P$  sends  $m_1$  to  $V$ , and  $V$  replies with a uniformly chosen bit  $b$ .  $P$  replies by sending  $m_2$  to  $V$ , and  $V$  accepts or rejects  $x$  based on  $\langle x, m_1, b, m_2 \rangle$ .*

Using the idea of [10] we show how to construct a NIC from a simulator  $S$  for any V-bit zero-knowledge protocol  $\langle P, V \rangle$ . The NIC will be hiding on YES instances, and binding on NO instances. We start with the following idea to commit to a bit  $b$ : use randomness  $r$  to execute  $S$  on input  $x$ , obtain a transcript  $\langle m_1, b', m_2 \rangle$  such that  $b = b'$  and  $V$  accepts, and output  $m_1$  as a commitment. If  $x$  is a YES instance, then the perfect completeness property guarantees that we always obtain transcripts where  $V$  accepts, and since  $b$  cannot be determined from such  $m_1$ , the commitment is hiding. Conversely, by the soundness of  $\langle P, V \rangle$ , if  $x$  is a NO instance, then there are no transcripts  $\langle m_1, 0, m_2 \rangle$  and  $\langle m_1, 1, m'_2 \rangle$  such that  $V$  accepts in both. The problem with this idea is that  $b'$  may not be equal to  $b$ . To overcome this issue we redefine the commitment to be  $\langle m_1, b' \oplus b \rangle$ . That is, we execute  $S(x)$ , obtain  $\langle m_1, b', m_2 \rangle$ , and output  $\langle m_1, b' \oplus b \rangle$ . Intuitively, since  $b'$  is hidden, the bit  $b' \oplus b$  is also hidden. Thus, the scheme is hiding. Our lemma follows.

**Lemma 3.1.** *Let  $\Pi = \langle \Pi_Y, \Pi_N \rangle$  be a promise-problem. If  $\Pi$  has a V-bit, public-coin **HVPZK** (respectively, **HVSZK**, **HVCZK**) proof, then  $\Pi$  has a NIC that is perfectly (respectively, statistically, computationally) hiding on  $\Pi_Y$  and perfectly binding on  $\Pi_N$ .*

*Proof.* Fix a public-coin, V-bit **HVPZK** (respectively, **HVSZK**, **HVCZK**) proof  $\langle P, V \rangle$  for  $\Pi$ , and fix a simulator  $S$  for  $\langle P, V \rangle$ . Without loss of generality we can assume that  $S$  either outputs transcripts in which  $V$  accepts, or it outputs `fail`. Using  $S$  we define a NIC  $f$  for  $\Pi$  as follows. Let  $f(x, b; r)$  be the function that executes  $S(x)$  with randomness  $r$ . If  $f$  obtains a transcript  $\langle x, m'_1, b', m'_2 \rangle$  such that  $V(x, m'_1, b', m'_2) = \text{accept}$ , then  $f$  outputs  $\langle m'_1, b' \oplus b \rangle$ . Otherwise,  $f$  outputs `fail`.

We show that  $f$  is binding on  $\Pi_N$ . Let  $x \in \Pi_N$ . Notice that for any  $r$  and  $b$  it holds that  $f(x, b; r)$  outputs one bit if and only if  $f(x, b; r) = b$ . Thus, if  $f$  outputs one bit, then there are no  $r$  and  $r'$  such that  $f(x, 0; r) = f(x, 1; r')$ . For the case where  $f(x, b; r)$  outputs a pair  $\langle \tilde{m}_1, \tilde{b} \rangle$ , recall that  $\tilde{b} = b' \oplus b$ , where  $b'$  is taken from some transcript  $\langle x, m'_1, b', m'_2 \rangle$ . Thus, by the definition of  $f$ , for any  $\tilde{m}_1, \tilde{b}, r$  and  $r'$  it holds that  $f(x, 0; r) = f(x, 1; r') = \langle \tilde{m}_1, \tilde{b} \rangle$  if and only if there are  $m_2$  and  $m'_2$  and

such that  $V(x, m_1, 0, m_2) = V(x, m_1, 1, m'_2) = \text{accept}$ . However,  $\langle P, V \rangle$  is public coin, and by the soundness property of  $\langle P, V \rangle$  there are no  $m_1, m_2$  and  $m'_2$  such that  $V(x, m_1, 0, m_2) = V(x, m_1, 1, m'_2) = \text{accept}$ . Hence, if  $f$  does not output one bit, then there are no  $r$  and  $r'$  such that  $f(x, 0; r) = f(x, 1; r')$ . We conclude that  $f$  is perfectly binding on  $\Pi_N$ .

The rest of the proof shows that  $f$  is hiding on  $\Pi_Y$ . Starting with the statistical setting, we calculate the statistical distance between commitments to 0 and commitments to 1 over  $x \in \Pi_Y$ . The following probabilities are over the randomness  $r$  for  $f$ .

$$\begin{aligned} \Delta(f(x, 0), f(x, 1)) &= \frac{1}{2} \sum_{\alpha} |\Pr[f(x, 0) = \alpha] - \Pr[f(x, 1) = \alpha]| \\ &= \frac{1}{2} \sum_{m_1} |\Pr[f(x, 0) = \langle m_1, 0 \rangle] - \Pr[f(x, 1) = \langle m_1, 0 \rangle]| + \\ &\quad \frac{1}{2} \sum_{m_1} |\Pr[f(x, 0) = \langle m_1, 1 \rangle] - \Pr[f(x, 1) = \langle m_1, 1 \rangle]| + \\ &\quad \frac{1}{2} \sum_b |\Pr[f(x, 0) = b] - \Pr[f(x, 1) = b]|. \end{aligned}$$

For any  $x$  we define  $p_x \stackrel{\text{def}}{=} \Pr[S(x) = \text{fail}]$ , where the probability is over the randomness to  $S$ . In addition, when  $S$  is a **HVPZK** simulator we are assuming that  $p_x = 0$ . By the definition of  $f$ , the above sum over  $b$  equals  $p_x$ . It remains to deal with the sums over  $m_1$ . We show that the first sum is upper bounded by  $\Delta(\langle P, V \rangle(x), S(x)) - p_x/2$ , and since a symmetric argument applies to the second sum, the total will be upper bounded by  $2 \cdot \Delta(\langle P, V \rangle(x), S(x))$ . The following probabilities for  $\langle P, V \rangle(x)$  and  $S(x)$  are over the randomness to  $P, V$  and  $S$ , respectively.

$$\begin{aligned} &\frac{1}{2} \sum_{m_1} |\Pr[f(x, 0) = \langle m_1, 0 \rangle] - \Pr[f(x, 1) = \langle m_1, 0 \rangle]| = \\ &\frac{1}{2} \sum_{m_1} \left| \sum_{m_2} \Pr[S(x) = \langle m_1, 0, m_2 \rangle] - \sum_{m_2} \Pr[S(x) = \langle m_1, 1, m_2 \rangle] \right| = \\ &\frac{1}{2} \sum_{m_1} \left| \sum_{m_2} \Pr[S(x) = \langle m_1, 0, m_2 \rangle] - \sum_{m_2} \Pr[\langle P, V \rangle(x) = \langle m_1, 0, m_2 \rangle] \right. \\ &\quad \left. - \left( \sum_{m_2} \Pr[S(x) = \langle m_1, 1, m_2 \rangle] - \sum_{m_2} \Pr[\langle P, V \rangle(x) = \langle m_1, 1, m_2 \rangle] \right) \right| \leq \\ &\frac{1}{2} \sum_{m_1, m_2} (|\Pr[S(x) = \langle m_1, 0, m_2 \rangle] - \Pr[\langle P, V \rangle(x) = \langle m_1, 0, m_2 \rangle]| + \\ &\quad |\Pr[S(x) = \langle m_1, 1, m_2 \rangle] - \Pr[\langle P, V \rangle(x) = \langle m_1, 1, m_2 \rangle]|) = \\ &\Delta(\langle P, V \rangle(x), S(x)) - p_x/2. \end{aligned}$$

Above we used the fact that  $S$  outputs transcripts in which  $V$  accepts, and then we used the fact that  $\langle P, V \rangle$  is public-coin (which implies that for any  $m_1$  the probability to choose an element of  $\langle P, V \rangle(x)$  whose prefix is  $\langle m_1, 0 \rangle$  equals the probability to choose an element of  $\langle P, V \rangle(x)$  whose prefix is  $\langle m_1, 1 \rangle$ ). We conclude that  $\Delta(f(x, 0), f(x, 1)) \leq 2 \cdot \Delta(S(x), \langle P, V \rangle(x))$ . Hence, if  $S$  is a **HVPZK** (respectively, **HVSZK**) simulator, then  $\Delta(S(x), \langle P, V \rangle(x))$  is 0 for any  $x \in \Pi_Y$  (respectively,

negligible on  $\Pi_Y$ ), which implies that  $f$  is perfectly (respectively, statistically) hiding on  $\Pi_Y$ .

It remains to deal with the case that  $S$  is a **HVCZK** simulator. The analysis is analogous to the statistical setting, but in reverse. We define the function  $f'(\cdot, b)$  just like  $f$ , except that instead of executing the simulator,  $f'$  receives a transcript  $\langle m_1, b', m_2 \rangle$  and outputs  $\langle m_1, b' \oplus b \rangle$ . Thus,  $f'(S(x), b)$  and  $f(x, b)$  are identically distributed for any  $b \in \{0, 1\}$ . Assume towards contradiction that there is a probabilistic, polynomial-time Turing machine  $D$  that distinguishes  $\{f(x, 0)\}_{x \in \Pi_Y}$  and  $\{f(x, 1)\}_{x \in \Pi_Y}$ . Thus,  $D$  distinguishes  $\{f'(S(x), 0)\}_{x \in \Pi_Y}$  and  $\{f'(S(x), 1)\}_{x \in \Pi_Y}$ , and the following expression is non-negligible:

$$\begin{aligned} & |\Pr[D(f'(S(x), 0)) = 1] - \Pr[D(f'(S(x), 1)) = 1]| \leq \\ & |\Pr[D(f'(S(x), 0)) = 1] - \Pr[D(f'(\langle P, V \rangle(x), 0)) = 1]| + \\ & |\Pr[D(f'(S(x), 1)) = 1] - \Pr[D(f'(\langle P, V \rangle(x), 1)) = 1]|. \end{aligned}$$

Above we used the fact that  $\langle P, V \rangle$  is V-bit, which implies that  $f'(\langle P, V \rangle(x), 0)$  and  $f'(\langle P, V \rangle(x), 1)$  are identically distributed for any  $x \in \Pi_Y$ . It follows that there is  $b \in \{0, 1\}$  such that  $D$  distinguishes  $\{f'(\langle P, V \rangle, b)\}_{x \in \Pi_Y}$  and  $\{f'(S(x), b)\}_{x \in \Pi_Y}$ . Since  $f'$  is efficient, this contradicts the fact that  $S$  is a **HVCZK** simulator. We conclude that  $f$  is computationally hiding on  $\Pi_Y$ . The lemma follows.

Theorem 3.1 presented in the beginning of this section immediately follows from Lemmas 2.1 and 3.1. Thus, we get a characterization of V-bit zero-knowledge protocols as NIC. We remark that Theorem 3.1 can be extended to arguments, and to relaxed notions of V-bit protocols.

## 4 Random Self-reducibility Implies NIC

We prove the folklore theorem that if a problem  $\Pi$  is random self-reducible, then  $\Pi$  has a perfectly hiding NIC. Our proof uses the idea behind the construction of the subroutine in the protocol of [25] (see Section 3.3 in [25]). Combining this theorem with our closure result from the next section allows us to strengthen and unify the results of [26, 25, 16], and achieve all the improvements claimed in the introduction. We define random self-reducibility.

**Definition 4.1 (Random self-reducible language [2]).** Let  $\mathcal{N} \subset \{0, 1\}^*$  be a countable set such that  $R_x$  is an NP-relation for each  $x \in \mathcal{N}$ . The domain of  $R_x$  is denoted  $d(R_x) \stackrel{\text{def}}{=} \{z \mid \exists w \langle z, w \rangle \in R_x\}$ . The language  $L \stackrel{\text{def}}{=} \{\langle x, z \rangle \mid x \in \mathcal{N}, \exists w \langle z, w \rangle \in R_x\}$  is random self-reducible (RSR) if there are polynomial time algorithms  $G, A_1, A_2$ , and  $S$  such that  $S(x, z; r) = y \in d(R_x)$  for any  $x \in \mathcal{N}, z$ , and  $r$ , and the following conditions hold.

1. If  $z \in d(R_x)$ , and  $r$  is uniformly distributed, then  $y$  is uniformly distributed in  $d(R_x)$ .
2. A witness for  $y$  yields a witness for  $z$ , and vice versa. That is,  $\langle z, A_1(x, y, r, w') \rangle \in R_x$  for any  $\langle y, w' \rangle \in R_x$ , and  $\langle y, A_2(x, z, r, w'') \rangle \in R_x$  for any  $\langle z, w'' \rangle \in R_x$ .



3.  $G(x; r) = \langle z', w' \rangle \in R_x$ , and if  $r$  is uniformly distributed, then  $z'$  is uniformly distributed in  $d(R_x)$ , and  $w'$  is uniformly distributed in  $\{w | \langle z, w \rangle \in R_x\}$ .

We prove that random self-reducible problems have a perfectly hiding NIC. Given  $\mathcal{N}$  and  $R_x$  as in Definition 4.1 we define the problem  $\Pi^L \stackrel{\text{def}}{=} \langle \Pi_Y^L, \Pi_N^L \rangle$ , where  $\Pi_Y^L \stackrel{\text{def}}{=} \{\langle x, z \rangle | x \in \mathcal{N}, \exists w \langle z, w \rangle \in R_x\}$ , and  $\Pi_N^L \stackrel{\text{def}}{=} \{\langle x, z \rangle | x \in \mathcal{N}, \forall w \langle z, w \rangle \notin R_x\}$ .

**Lemma 4.1.** *If  $L$  is a random self-reducible language, then  $\Pi^L$  has a perfectly hiding NIC.*

*Proof.* Let  $L \stackrel{\text{def}}{=} \{\langle x, z \rangle | x \in \mathcal{N}, \exists w \langle z, w \rangle \in R_x\}$  be a random self-reducible language. Consider the algorithms  $S$  and  $G$  from Definition 4.1. Let  $G'(x; r)$  be the algorithm that executes  $G(x; r)$ , obtains  $\langle z', w' \rangle$ , and outputs  $z'$ . We use  $S$  and  $G'$  to commit to 0 and 1, respectively. Formally, we define our NIC to be the probabilistic, polynomial-time Turing machine  $f(x, z, b; r)$  that on input  $\langle x, z \rangle \in \Pi_Y^L \cup \Pi_N^L$ , bit  $b$ , and randomness  $r$  outputs  $S(x, z; r)$  if  $b = 0$ , and  $G'(x; r)$  if  $b = 1$ .

The efficiency of  $f$  follows from the efficiency of  $S$  and  $G$ . We show that  $f$  is perfectly hiding. By Definition 4.1,  $S(x, z; r) = y$  is uniformly distributed over  $d(R_x)$  if  $r$  is uniformly distributed, and  $\langle x, z \rangle \in \Pi_Y^L$ . Similarly,  $G'(x; r) = \langle z', w' \rangle$ , and  $z'$  is uniformly distributed over  $d(R_x)$  if  $r$  is uniformly distributed and  $x \in \mathcal{N}$ . Since the output of  $f$  is uniformly distributed over  $d(R_x)$  for any  $b$  and  $\langle x, z \rangle \in \Pi_Y^L$ , the ensembles  $\{f(x, z, 0; r)\}_{\langle x, z \rangle \in \Pi_Y^L}$  and  $\{f(x, z, 1; r)\}_{\langle x, z \rangle \in \Pi_Y^L}$  are statistically identical, and therefore  $f$  is perfectly hiding on  $\Pi_Y^L$ .

We show that  $f$  is binding on  $\Pi_N^L$ . Let  $\langle x, z \rangle \in \Pi_N^L$ . Assume towards contradiction that there are  $r$  and  $r'$  such that  $S(x, z; r) = f(x, z, 0; r) = f(x, z, 1; r') = G'(x; r)$ . Let  $y = S(x, z; r)$ . By the definition of  $G'$ , there is  $w'$  such that  $G(x; r) = \langle G'(x; r), w' \rangle = \langle y, w' \rangle \in R_x$ . By the property of  $A_1$  from Definition 4.1, it follows that  $\langle z, A_1(x, y, r, w') \rangle \in R_x$ . Hence,  $\langle x, z \rangle \in \Pi_Y^L$ , in contradiction to the choice of  $\langle x, z \rangle \in \Pi_N^L$ . Thus,  $f$  is binding on  $\Pi_N^L$ .

Notice that in the above proof we did not use Algorithm  $A_2$  from Definition 4.1. Neither did we use the fact that  $A_1$  runs in polynomial time, nor did we use the witness outputted by  $G$ .

## 5 Closure of Problems Possessing NIC Under Monotone Boolean Formulae

We use the technique of [25] to show that the class of problems possessing NIC is closed under *arbitrary* (as opposed to fixed) monotone boolean formulae. For perfectly hiding NIC the analysis is simple, but for statistically and computationally hiding NIC the analysis is more complicated.

**Motivation.** Let  $f$  be a perfectly hiding NIC for a problem  $\Pi$ . Consider a prover and a verifier who are given instances  $x_0, \dots, x_n \in \Pi_Y \cup \Pi_N$ , and suppose that the prover wants to prove to the verifier that more than half of the  $x_i$ 's are in  $\Pi_Y$ . This statement can be expressed using the logical connectors AND (denoted  $\wedge$ ) and OR (denoted  $\vee$ ). The

prover can prove this statement if we can construct a NIC  $f'$  that is hiding when more than half of the  $x_i$  are in  $\Pi_Y$ , and binding otherwise. This is so because the statement is an **NP** statement, and the prover can use  $f'$  in the protocol of Blum [5] (as in Section 2). Later we will give a general construction that yields such  $f'$ . For now we consider the simple case where  $n = 2$ . That is, the prover proves that both  $x_0$  and  $x_1$  are in  $\Pi_Y$ .

To formulate the fact that the statement being proved is  $x_0 \in \Pi_Y \wedge x_1 \in \Pi_Y$  we define the common input as  $\langle \phi, x_0, x_1 \rangle$ , where  $\phi = a \wedge b$ . Recall that we want to use the NIC  $f$  for  $\Pi$  to construct a NIC  $f'$  which is hiding when  $x_0 \in \Pi_Y \wedge x_1 \in \Pi_Y$ , and binding otherwise. We can construct such  $f$  by defining  $f'(x_0, x_1, b) \stackrel{\text{def}}{=} \langle f(x_0, b), f(x_1, b) \rangle$ . Thus, if  $x_0, x_1 \in \Pi_Y$ , then both  $f(x_0, b)$  and  $f(x_1, b)$  hide  $b$ , which implies that  $f'$  is hiding, and if  $x_i \in \Pi_N$  (for some  $i \in \{0, 1\}$ ), then  $f(x_i, b)$  binds to  $b$ , and  $f'$  is binding. Notice that we omitted the randomness of  $f'$  from the notation, but the intention is that  $f'$  uses independent randomness in each execution of  $f$ .

We can formulate other statements too. For example, consider a prover and a verifier who are given  $x_0, x_1$ , and the prover wants to prove that either  $x_0 \in \Pi_Y$  or  $x_1 \in \Pi_Y$ . Again, we can formulate this statement by defining  $\langle \phi, x_0, x_1 \rangle$  as the input, where  $\phi = a \vee b$ . Recall that we want to use the NIC  $f$  for  $\Pi$  to construct a NIC  $f'$  which is hiding when  $x_0 \in \Pi_Y \vee x_1 \in \Pi_Y$ , and hiding otherwise. We can construct such  $f$  by defining  $f'(x_0, x_1, b) \stackrel{\text{def}}{=} \langle f(x_0, b_0), f(x_1, b_1) \rangle$ , where  $b_0$  is uniformly chosen, and  $b_1$  is chosen such that  $b_0 \oplus b_1 = b$ . Thus, if  $x_0, x_1 \in \Pi_N$ , then both  $f(x_0, b)$  and  $f(x_1, b)$  bind to  $b$ , which implies that  $f'$  is binding, and if  $x_i \in \Pi_Y$  (for some  $i \in \{0, 1\}$ ), then  $f(x_i, b)$  hides  $b_i$ , and thus  $f$  hides  $b$ . Based on these  $\wedge$  and  $\vee$  cases we can give a general construction of a NIC  $f'$  from a NIC  $f$ .

**Construction 5.1.** *Let  $f$  be a NIC, and let  $b \in \{0, 1\}$ . Let  $\phi$  be a monotone boolean formula over the variables  $a_1, \dots, a_m$ , and let  $\vec{x} = \langle x_1, \dots, x_n \rangle$  be a vector of  $n$  strings, where  $n \geq m$ . Let  $r \in \{0, 1\}^*$  be a uniformly distributed input to  $f'$ .*

*The recursive function  $f'(\phi, \vec{x}, f, b; r)$  is defined as follows.*

1. *If  $\phi = a_i$  for some  $1 \leq i \leq m$ , then return  $f(x_i, b, r)$ .*
2. *Otherwise, there are monotone boolean formulae  $\phi_0$  and  $\phi_1$  such that  $\phi = \phi_0 \wedge \phi_1$  or  $\phi = \phi_0 \vee \phi_1$ . Partition  $r$  into  $r_0$  and  $r_1$ .*
3. *If  $\phi = \phi_0 \wedge \phi_1$ , then return  $\langle f'(\phi_0, \vec{x}, f, b, r_0), f'(\phi_1, \vec{x}, f, b, r_1) \rangle$ .*
4. *If  $\phi = \phi_0 \vee \phi_1$ , then return  $\langle f'(\phi_0, \vec{x}, f, b_0, r_0), f'(\phi_1, \vec{x}, f, b_1, r_1) \rangle$ , where  $b_0 \in \{0, 1\}$  is uniformly distributed, and  $b_1$  is chosen such that  $b_0 \oplus b_1 = b$ .*

Our next step is define a problem that allows the prover to prove *arbitrary* (as opposed to fixed) monotone, boolean formula statements. We need the following definitions. A *boolean variable* is a variable that can only take the values 0 or 1. We say that  $\phi$  is a monotone boolean formula if  $\phi$  is a boolean variable, or  $\phi$  is  $\phi_0 \wedge \phi_1$  or  $\phi_0 \vee \phi_1$ , where both  $\phi_0$  and  $\phi_1$  are monotone boolean formulae. Let  $\Pi = \langle \Pi_Y, \Pi_N \rangle$  be a promise-problem, and let  $x \in \Pi_Y \cup \Pi_N$ . The *characteristic function*  $\chi_\Pi$  of  $\Pi$  is defined as follows: if  $x \in \Pi_Y$ , then  $\chi_\Pi(x) = 1$ , and if  $x \in \Pi_N$ , then  $\chi_\Pi(x) = 0$ . Let  $\phi$  be a boolean formula over  $a_1, \dots, a_m$ , and let  $x_1, \dots, x_n \in \Pi_Y \cup \Pi_N$  for some  $n \geq m$ . The *evaluation* of  $\phi$  in  $\vec{x} = \langle x_1, \dots, x_n \rangle$  is denoted  $\phi(\vec{x})$ , and equals 1 if and only if  $\phi$  is satisfied when  $a_i$  is assigned  $\chi_\Pi(x_i)$  for each  $1 \leq i \leq m$ .

We say that a class  $C$  of problems is closed under *arbitrary*, monotone boolean formulae if  $\Pi \in C$  implies that  $\Phi(\Pi) \in C$ , where  $\Phi(\Pi)$  is defined as follows.

**Definition 5.1.** Let  $\Pi = \langle \Pi_Y, \Pi_N \rangle$  be a problem. The problem  $\Phi(\Pi) \stackrel{\text{def}}{=} \langle \Phi(\Pi)_Y, \Phi(\Pi)_N \rangle$  is defined as

$$\Phi(\Pi)_Y \stackrel{\text{def}}{=} \{ \langle \phi, x_1, \dots, x_n \rangle \mid \phi(\chi_\Pi(x_1), \dots, \chi_\Pi(x_n)) = 1 \}$$

$$\Phi(\Pi)_N \stackrel{\text{def}}{=} \{ \langle \phi, x_1, \dots, x_n \rangle \mid \phi(\chi_\Pi(x_1), \dots, \chi_\Pi(x_n)) = 0 \},$$

where  $\phi$  is a monotone boolean formula over  $a_1, \dots, a_m$  such that  $m \leq n$ , and  $x_i \in \Pi_Y \cup \Pi_N$  for all  $1 \leq i \leq n$ . We define  $\Phi(\Pi)^k \stackrel{\text{def}}{=} \langle \Phi(\Pi)_Y^k, \Phi(\Pi)_N \rangle$ , where  $\Phi(\Pi)_Y^k$  is defined as

$$\Phi(\Pi)_Y^k \stackrel{\text{def}}{=} \{ \langle \phi, x_1, \dots, x_n \rangle \mid \phi(\chi_\Pi(x_1), \dots, \chi_\Pi(x_n)) = 1 \wedge \forall i \mid x_i \mid^k \geq \mid \phi, x_1, \dots, x_n \mid \}.$$

The definition of  $\Phi(\Pi)$  allows the prover to prove *arbitrary* (as opposed to fixed) monotone, boolean formula statements, and so does the definition of  $\Phi(\Pi)^k$ . This formulation has the advantage that the formula does not have to be hardwired into the protocol, or known in advance. Our theorem follows.

**Theorem 5.2.** Let  $\Pi = \langle \Pi_Y, \Pi_N \rangle$  be a promise-problem with a NIC  $f$ , and let  $f'$  be the function constructed from  $f$ , given in Construction 5.1. Let  $k \in \mathbb{N}$ .

1. If  $f$  is a perfectly hiding NIC for  $\Pi$ , then  $f'$  is a perfectly hiding NIC for  $\Phi(\Pi)$ .
2. If  $f$  is a statistically (respectively, computationally) hiding NIC for  $\Pi$ , then  $f'$  is a statistically (respectively, computationally) hiding NIC for  $\Phi(\Pi)^k$ .

## References

1. Aiello, W., Håstad, J.: Statistical zero-knowledge languages can be recognized in two rounds. *J. of Computer and System Sciences* 42(3), 327–345 (1991)
2. Angluin, D., Lichtenstein, D.: Provable security in cryptosystems: a survey. Technical Report 288, Department of Computer Science, Yale University (1983)
3. Barak, B.: How to go beyond the black-box simulation barrier. In: FOCS, pp. 106–115 (2001)
4. Bellare, M., Micali, S., Ostrovsky, R.: Perfect zero-knowledge in constant rounds. In: 22nd STOC, pp. 482–493 (1990)
5. Blum, M.: How to prove a theorem so no one else can claim it. In: Proceedings of the ICM, pp. 1444–1451 (1986)
6. Boppana, R.B., Håstad, J., Zachos, S.: Does co-NP have short interactive proofs? *Inf. Process. Lett.* 25(2), 127–132 (1987)
7. Cramer, R.: Modular Design of Secure yet Practical Cryptographic Protocols. PhD thesis, CWI and Uni. of Amsterdam (1996)
8. Cramer, R., Damgård, I., MacKenzie, P.D.: Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Public Key Cryptography, pp. 354–372 (2000)
9. Dâmgård, I., Cramer, R.: On monotone function closure of perfect and statistical zero-knowledge (1996)
10. Damgård, I.B.: On the existence of bit commitment schemes and zero-knowledge proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 17–27. Springer, Heidelberg (1990)

11. Damgård, I.B.: On  $\Sigma$ -protocols (2005) Available online at [www.daimi.au.dk/~ivan/Sigma.pdf](http://www.daimi.au.dk/~ivan/Sigma.pdf)
12. Fortnow, L.: The complexity of perfect zero-knowledge. In: Micali, S. (ed.) *Advances in Computing Research*, vol. 5, pp. 327–343. JAC Press (1989)
13. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM* 38(3), 691–729 (1991)
14. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* 18(1), 186–208 (1989)
15. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. *SIAM J. Comput.* 28(4), 1364–1396 (1999)
16. Itoh, T., Ohta, Y., Shizuya, H.: A language-dependent cryptographic primitive. *J. Cryptology* 10(1), 37–50 (1997)
17. Micali, S., Pass, R.: Local zero knowledge. In: *STOC*, pp. 306–315 (2006)
18. Micciancio, D., Ong, S.J., Sahai, A., Vadhan, S.P.: Concurrent zero knowledge without complexity assumptions. In: Halevi, S., Rabin, T. (eds.) *TCC 2006*. LNCS, vol. 3876, pp. 1–20. Springer, Heidelberg (2006)
19. Micciancio, D., Vadhan, S.P.: Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In: Boneh, D. (ed.) *CRYPTO 2003*. LNCS, vol. 2729, pp. 282–298. Springer, Heidelberg (2003)
20. Naor, M.: Bit commitment using pseudorandomness. *J. Cryptology* 4(2), 151–158 (1991)
21. Nguyen, M.-H., Vadhan, S.: Zero knowledge with efficient provers. In: *STOC '06*. Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, Seattle, WA, USA, pp. 287–295. ACM Press, New York (2006)
22. Ong, S.J., Vadhan, S.: Zero knowledge and soundness are symmetric. *Electronic Colloquium on Computational Complexity (ECCC)* (TR06-139) (2006)
23. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: *FOCS*, pp. 366–375 (2002)
24. Sahai, A., Vadhan, S.P.: A complete problem for statistical zero-knowledge. *J. ACM* 50(2), 196–249 (2003)
25. De Santis, A., Di Crescenzo, G., Persiano, G., Yung, M.: On monotone formula closure of SZK. In: *IEEE Symposium on Foundations of Computer Science*, pp. 454–465. IEEE Computer Society Press, Los Alamitos (1994)
26. Tompa, M., Woll, H.: Random self-reducibility and zero-knowledge interactive proofs of possession of information. In: *28th FOCS*, pp. 472–482 (1987)
27. Vadhan, S.P.: An unconditional study of computational zero knowledge. In: *FOCS*, pp. 176–185 (2004)
28. Watrous, J.: Zero-knowledge against quantum attacks. In: *STOC*, pp. 296–305 (2006)

# Sharp Tractability Borderlines for Finding Connected Motifs in Vertex-Colored Graphs

Michael R. Fellows<sup>1,2,\*</sup>, Guillaume Fertin<sup>3</sup>,  
Danny Hermelin<sup>4,\*\*</sup>, and Stéphane Vialette<sup>5</sup>

<sup>1</sup> The University of Newcastle, Callaghan NSW 2308 - Australia  
`mike.fellows@cs.newcastle.edu.au`

<sup>2</sup> Institute of Advanced Study, Durham University,  
Durham DH1 3RL - United Kingdom

<sup>3</sup> Laboratoire d'Informatique de Nantes-Atlantique (LINA), FRE CNRS 2729  
Université de Nantes, 2 rue de la Houssinière, 44322 Nantes Cedex 3 - France  
`fertin@lina.univ-nantes.fr`

<sup>4</sup> Department of Computer Science, University of Haifa,  
Mount Carmel, Haifa 31905 - Israel  
`danny@cri.haifa.ac.il`

<sup>5</sup> Laboratoire de Recherche en Informatique (LRI), UMR CNRS 8623  
Faculté des Sciences d'Orsay - Université Paris-Sud, 91405 Orsay - France  
`vialette@lri.fr`

**Abstract.** We study the problem of finding occurrences of motifs in vertex-colored graphs, where a motif is a multiset of colors, and an occurrence of a motif is a subset of connected vertices whose multiset of colors equals the motif. This problem has applications in metabolic network analysis, an important area in bioinformatics. We give two positive results and three negative results that together draw sharp borderlines between tractable and intractable instances of the problem.

## 1 Introduction

Vertex-colored graph problems have numerous applications in bioinformatics. Sandwich problems have applications in DNA physical mapping [6,13,15] and in perfect phylogeny [8,19], while vertex-recoloring problems arise in protein-protein interaction networks and phylogenetic analysis [7,9,20]. In this paper, we consider another natural vertex-colored graph problem with an interesting application in bioinformatics:

---

\* This research has been supported by the Australian Research Council through the Australian Center for Bioinformatics, by the University of Newcastle Parameterized Complexity Research Unit under the auspices of the Deputy Vice-Chancellor for Research, and by a Fellowship to the Durham University Institute for Advanced Studies. The authors also gratefully acknowledge the support and kind hospitality provided by a William Best Fellowship at Grey College while the paper was in preparation.

\*\* Partially supported by the Caesarea Rothschild Institute (CRI).

## GRAPH MOTIF:

*Input:* A vertex-colored graph  $G$  and a multiset of colors  $M$ .

*Question:* Does  $G$  have a connected subset of vertices whose multiset of colors equals  $M$ ?

The GRAPH MOTIF problem was introduced in a slightly more general form by Lacroix, Fernandes, and Sagot (who allowed multiple colors per vertex) in the context of metabolic network analysis, an important area in bioinformatics [18]. There, vertices correspond to chemical compounds or reactions, and edges correspond to interactions between these compounds and reactions. The vertex coloring is used to distinguish between different types of chemicals and reactions. The transmission of information in these networks can usually be described as a chain of interacting chemicals, in which chemical interactions enable each chemical in the path to modify its successor so as to transmit biological data. In this scenario, connected motifs can correspond to relatively functional independent modules of the network which consist of a specific set of chemical compounds and reactions. It is argued in [18] that a method for a rational decomposition of a metabolic network into relatively independent functional subsets is essential for a better understanding of the modularity and organization principles in the network. We refer the reader to [11,18] for more biological background of the problem. We also refer to [16,17] for related work and relevant background.

In [18], GRAPH MOTIF is proved to be NP-complete even if the given vertex-colored graph is a tree, but fixed-parameter tractable in this case when parameterized by the size of the given motif (*i.e.*  $|M|$ ). However, as observed by [18], their fixed-parameter does not apply when the vertex-colored graph is a general graph. For this case they only provided a heuristic algorithm which works well in practice. This motivates us to further investigate the tractability landscape of GRAPH MOTIF, and in particular, to investigate it under different parameters which govern the structure of its input. We give an extensive analysis for GRAPH MOTIF, applying techniques from both classical and parameterized complexity, that unravels sharp borderlines between tractable and intractable instances of the problem. More specifically, we give two algorithms and three hardness results that together imply:

1. For motifs of unbounded size, GRAPH MOTIF is NP-complete already for trees of maximum degree 3, even if the motif is a set of colors rather than a multiset. For motifs of logarithmic size (in the number of vertices of  $G$ ), the problem is polynomial-time solvable in any general graph.
2. GRAPH MOTIF is NP-complete for motifs with 2 colors, even if  $G$  is bipartite with maximum degree 4. However, it is polynomial-time solvable in constant treewidth graphs for motifs consisting of any constant number of colors (and arbitrary size). When the number of colors in the motif is taken as a parameter, GRAPH MOTIF is W[1]-hard even in case  $G$  is a tree.

The rest of the paper is organized as follows. In the remainder of this section we discuss notations that will be used throughout the paper. In Section 2, we give

two NP-hardness results that will motivate the rest of our discussion. Following this, in Section 3 we present a fixed-parameter algorithm (parameterized by  $|M|$ ) that applies to any general graph. In Section 4 we discuss the case when  $G$  has bounded treewidth. Finally, in Section 5, we show that GRAPH MOTIF is  $W[1]$ -hard on trees when parameterized by the number of colors in  $M$ .

**Basic notation and terminology:** Throughout the paper, we use  $G = (V(G), E(G))$  to denote our given vertex-colored graph, and  $n = |V(G)|$  to denote its order. For a vertex  $v \in V(G)$ , we use  $\chi(v)$  to denote the color of  $v$ , and for a vertex subset  $V \subseteq V(G)$ , we let  $\chi(V)$  denote the multiset of colors  $\bigcup_{v \in V} \chi(v)$ . For any vertex subset  $V \subseteq V(G)$ , we let  $G[V]$  denote the subgraph of  $G$  induced by  $V$ , *i.e.* the subgraph on  $V$  along with all edges of  $G$  that connect vertices in  $V$ . We assume w.l.o.g. that  $G$  is connected.

A motif  $M$  is a multiset of colors. If  $M$  is in fact a set rather than a multiset, we say that  $M$  is *colorful*. Given a subset of vertices  $V \subseteq V(G)$ ,  $|V| = |M|$ , we say that  $V$  is *colored by the colors of  $M$* , if  $\chi(V) = M$ . For  $V$  to be an *occurrence* of  $M$ , we require not only for  $V$  to be colored by the colors of  $M$ , but also for  $G[V]$  to be connected. If this is in fact the case, we say that  $M$  *occurs at  $v$*  for any vertex  $v \in V$ . In these terms, the GRAPH MOTIF problem is the problem of determining whether a given motif  $M$  occurs at any vertex of a given vertex-colored graph  $G$ . We assume w.l.o.g. that  $\chi(v) \in M$  for any  $v \in V(G)$ .

Our analysis is based both on the classical and parameterized complexity frameworks. Readers unfamiliar with these subjects are referred to [12, 14].

## 2 Tight NP-Hardness Results

As mentioned previously in Section 1, GRAPH MOTIF is already known to be NP-complete for trees in [18]. Our aim in this section is to tighten this result by showing that GRAPH MOTIF remains hard for highly restrictive graph classes, even if we restrict ourselves to motifs which are sets rather than multisets, or to motifs which consist of a small number of colors.

We first consider colorful motifs. Recall that a motif  $M$  is colorful if it consists of  $|M|$  distinct colors. At first sight, it might seem that occurrences of colorful motifs should be easier to find, at least for certain types of graphs. Unfortunately, the following theorem proves that this is apparently not the case.

**Theorem 1.** GRAPH MOTIF is NP-complete, even if  $M$  is colorful and  $G$  is a tree of maximum degree three.

*Proof.* GRAPH MOTIF is clearly in NP. To prove NP-hardness, we present a reduction from the well known NP-complete problem 3-SAT [14]. Recall that 3-SAT asks to determine whether a given 3-CNF formula is satisfiable, that is, whether there is a truth assignment to the boolean variables of the formula, such that the value of the formula under this assignment is 1. The problem remains hard even if each variable appears in at most three clauses and each literal (*i.e.* variable with or without negation) appears in at most two clauses [14]. Hence, we restrict ourselves in our proof to formulas of this type.

Let an instance of 3-SAT be given in the form of a 3-CNF formula  $\Phi = c_1 \wedge \dots \wedge c_m$  over variables  $x_1, \dots, x_n$  such that  $|\{c_j \mid x_i \in c_j\}| \leq 2$  and  $|\{c_j \mid \bar{x}_i \in c_j\}| \leq 2$  for all  $1 \leq i \leq n$ . We construct an instance for GRAPH MOTIF as follows. The colored graph  $G$  initially consists of a path of  $n$  vertices, each colored by a distinct color in  $1, \dots, n$ . To a vertex colored  $i$  in this path,  $1 \leq i \leq n$ , we connect a new vertex colored  $i'$ . To a vertex colored  $i'$ ,  $1 \leq i \leq n$ , we connect a pair of new non-adjacent vertices, both colored  $x_i$ . Conceptually, each vertex in this pair corresponds to a different truth assignment for  $x_i$ . If a truth assignment to variable  $x_i$  satisfies clause  $c_j$ , we connect a new vertex colored  $c_j$  to the vertex colored  $x_i$  which corresponds to this assignment. This is done for every  $x_i \in \{x_1, \dots, x_n\}$  and every  $c_j \in \{c_1, \dots, c_m\}$ . We conclude our construction by specifying  $M$  to be the set of colors  $\{1, \dots, n, 1', \dots, n', x_1, \dots, x_n, c_1, \dots, c_m\}$ . Note that  $G$  and  $M$  are as required by the theorem.

The construction above is clearly polynomial. Hence, to complete the proof, we are left to show that  $M$  occurs in  $G$  if and only if  $\Phi$  is satisfiable. For the first direction, assume that there exists a truth assignment  $\phi$  which satisfies  $\Phi$ . Let  $N \subseteq V(G)$  be the subset of vertices in  $G$  which are colored by the colors in  $\{1, \dots, n, 1', \dots, n'\}$ , and let  $X \subseteq V(G)$  be the subset of vertices which correspond to assignment  $\phi$ . Hence,  $X$  consists of  $n$  vertices which are colored by the colors in  $\{x_1, \dots, x_n\}$ , and  $N \cup X$  induces a connected subgraph. Since  $\phi$  satisfies every clause in  $\Phi$ , by construction of  $G$  there is a vertex colored  $c_j$  in the neighborhood of  $X$  for every  $1 \leq j \leq m$ . In other words, there exists a subset  $C$  of neighbors of vertices in  $X$  which is colored by the colors in  $\{c_1, \dots, c_m\}$ . It follows that  $V = N \cup X \cup C$  is connected and is colored by the colors of  $M$ , and therefore is an occurrence of  $M$  in  $G$ .

For the converse direction, assume there exists an occurrence  $V$  of  $M$  in  $G$ . Let  $X \subseteq V$  be the vertices colored by the colors in  $\{x_1, \dots, x_n\}$ ,  $C \subseteq V$  be the vertices colored by the colors in  $\{c_1, \dots, c_m\}$ , and  $\phi$  the truth assignment corresponding to  $X$ . By construction, a vertex colored  $c_j$  is connected in  $G$  to a vertex colored  $x_i$  if and only if the truth assignment corresponding to this vertex satisfies clause  $c_j$ . Since  $C$  contains all colors in  $\{c_1, \dots, c_m\}$ , and since vertices in  $C$  are connected only to vertices in  $X$ , it follows that  $\phi$  satisfies every clause in  $\Phi$ , and so it satisfies  $\Phi$  itself.  $\square$

Theorem [1](#) implies that for motifs of unbounded cardinality, there are not many interesting graph classes left for which GRAPH MOTIF becomes polynomial-time solvable. Note that if  $G$  is a tree of maximum degree two, then  $G$  is actually a path, and GRAPH MOTIF becomes trivial (simply search through all subpaths of length  $|M|$ ). However, the motif in the construction above is not only of unbounded size, it also consists of an unbounded number of colors. One might hope that for motifs which consist of only a small number of colors, GRAPH MOTIF would become polynomial-time solvable. The following theorem shows that this is not the case in a very sharp sense.

**Theorem 2.** GRAPH MOTIF is NP-complete, even if  $M$  consists of two colors, and  $G$  is bipartite with maximum degree four.



*Proof.* We reduce from the EXACT COVER BY 3-SETS (X3C) problem, which is known to be NP-complete [14]. Recall that, given a set  $X = \{x_1, x_2, \dots, x_{3q}\}$  and a collection  $S = \{s_1, s_2, \dots, s_n\}$  of 3-element subsets of  $X$ , the X3C problem asks to determine whether there exists an exact cover of  $X$  in  $S$ , *i.e.* a sub-collection  $C \subseteq S$  such that every element of  $X$  is included in exactly one subset  $s_i \in C$ . The problem is hard even if each element of  $X$  appears in at most three sets of  $S$  [14], so we restrict ourselves in the proof to instances of this type.

Let  $\langle X, S \rangle$  be an arbitrary instance of the X3C problem with  $|\{s_j \in S \mid x_i \in s_j\}| \leq 3$  for all  $x_i \in X$ . We show how to construct a motif  $M$  and a colored graph  $G$  in such a way that there exists an exact cover of  $X$  in  $S$  if and only if  $M$  occurs in  $G$ . First, we define  $M$  so as it contains  $2n + 3q$  *white* elements and  $q$  *black* elements. Then, we define  $G$  by  $V(G) = X \cup S \cup S' \cup S''$  and  $E(G) = E_1 \cup E_2 \cup E_3 \cup E_4$ , where  $S' = \{s'_1, s'_2, \dots, s'_n\}$  and  $S'' = \{s''_1, s''_2, \dots, s''_n\}$  are dummy copies of  $S$ , and  $E_1, E_2, E_3, E_4$  are defined by:  $E_1 = \{\{x_i, s_j\} \mid x_i \in s_j\}$ ,  $E_2 = \{\{s_i, s'_i\} \mid 1 \leq i \leq n\}$ ,  $E_3 = \{\{s'_i, s''_i\} \mid 1 \leq i \leq n\}$ , and  $E_4 = \{\{s'_i, s'_{i+1}\} \mid 1 \leq i \leq n - 1\}$ . The vertices of  $X \cup S' \cup S''$  are colored white and the vertices of  $S$  are colored black. It is easily seen that  $G$  and  $M$  are as required by the theorem, and that our construction can be carried out in polynomial time.

Let us now argue that there exists an exact cover  $C \subseteq S$  of  $X$  if and only if  $M$  occurs in  $G$ . For the first direction, suppose that there exists an exact cover  $C \subseteq S$  of  $X$ . Consider the subset of vertices  $V = X \cup C \cup S' \cup S''$ . First note that  $V$  consists of  $q = |C|$  black vertices and  $2n + 3q = |X \cup S' \cup S''|$  white vertices. Second, since  $C$  is a cover of  $X$ , every vertex of  $X$  is connected to some vertex in  $C$ , and  $C$  is connected to  $S' \cup S''$ , so  $V$  itself is connected. It follows that  $V$  is an occurrence of  $M$ , and  $M$  occurs in  $G$ .

Conversely, suppose that there exists an occurrence  $V \subseteq V(G)$  of  $M$  in  $G$ . Observe that  $M$  contains  $2n + 3q$  white elements, and since exactly  $2n + 3q$  vertices of  $G$  are colored white, we must have  $X \cup S' \cup S'' \subset V$ . The remaining  $q$  vertices in  $V$  are  $q$  black vertices from  $S$ . By construction, we do not have an edge between two vertices of  $X$ , nor between a vertex of  $X$  and a vertex of  $S' \cup S''$ . Therefore, since  $V$  is connected, each vertex of  $X$  has to be adjacent to at least one vertex in  $V \cap S$ . But  $|X| = 3q$  and each vertex in  $S$  is connected to exactly 3 vertices in  $X$ . Then it follows that no two vertices of  $V \cap S$  share a common neighbor in  $X$ , and  $C = V \cap S$  is an exact cover of  $X$  in  $S$ .  $\square$

### 3 A General Fixed-Parameter Algorithm

We now turn to show that GRAPH MOTIF is fixed-parameter tractable for parameter  $k = |M|$  on any general graph. More specifically, we present an  $\mathcal{O}(2^{\mathcal{O}(k)} n^2 \lg n)$  algorithm for the problem, which implies that GRAPH MOTIF for motifs of  $\mathcal{O}(\lg n)$  size is polynomial-time solvable. This is in striking contrast to the sharp hardness results given in the previous section. Our algorithm is based on the *color-coding technique* introduced by Alon *et al.* [2], whose derandomized version crucially relies on the notion of perfect hash families:

**Definition 1 (Perfect Hash Family).** *A family  $\mathcal{F}$  of functions from  $V(G)$  to  $\{1, \dots, k\}$  is perfect if for any subset  $V \subseteq V(G)$  of  $k$  vertices there is a function  $f \in \mathcal{F}$  which is one-to-one on  $V$ .*

Suppose  $M$  has an occurrence  $V$  in  $G$ , and suppose we are provided with a perfect family  $\mathcal{F}$  of functions from  $V(G)$  to  $\{1, \dots, k\}$ . Since  $\mathcal{F}$  is perfect, we are guaranteed that at least one function in  $\mathcal{F}$  assigns  $V$  with  $k$  distinct labels. Let  $f \in \mathcal{F}$  be such a function. For a given  $L \subseteq \{1, \dots, k\}$ , we define  $\mathcal{M}_L(v)$  to be the family of all motifs  $M' \subseteq M$ ,  $|M'| = |L|$ , for which there exists an occurrence  $V'$  with  $v \in V'$ , such that the set of (unique) labels that  $f$  assigns to  $V'$  is exactly  $L$ . Since  $M$  occurs in  $G$ , we know that  $M \in \mathcal{M}_{\{1, \dots, k\}}(v)$  for some  $v \in V(G)$ . Hence, to determine whether  $M$  occurs in  $G$ , we apply dynamic programming to compute  $\mathcal{M}_L(v)$  for all  $v \in V(G)$  and  $L \subseteq \{1, \dots, k\}$ .

Fix  $L$  to be some subset of  $\{1, \dots, k\}$ , and let  $v$  be any vertex of  $G$ . Our goal is to compute  $\mathcal{M}_L(v)$  assuming  $\mathcal{M}_{L'}(u)$  has been previously computed for every vertex  $u \in V(G)$  and any  $L' \subseteq L \setminus \{f(v)\}$ . The straightforward approach is to consider small motifs occurring at neighbors of  $v$ . However, a motif occurring at  $v$  might be the union of motifs occurring at any number of neighbors of  $v$ , and so this approach might require exponential running time in  $n$ . We therefore present an alternative method for computing  $\mathcal{M}_L(v)$ , which we call the *batch procedure*, that uses an even more naive approach, but one that requires exponential-time only with respect to  $k$ . Notice that while the motifs computed by the batch procedure are in general multisets of colors, the batch procedure always considers sets of distinct labels.

**Batch Procedure( $L, v$ )**

- Define  $\mathcal{M}$  to be the family of all pairs  $(M', L')$  such that  $M' \subseteq M \setminus \{\chi(v)\}$ ,  $L' \subseteq L \setminus \{f(v)\}$ , and  $M' \in \mathcal{M}_{L'}(u)$  for some  $u \in N(v)$ .
- Run through all pairs of  $(M', L'), (M'', L'') \in \mathcal{M}$  and determine whether  $M' \cup M'' \subseteq M \setminus \{\chi(v)\}$ , and whether  $L' \cap L'' = \emptyset$ . If there is such a pair, add  $(M' \cup M'', L' \cup L'')$  to  $\mathcal{M}$  and repeat this step. Otherwise, continue to the next step.
- Set  $\mathcal{M}_L(v)$  to be all motifs  $M' \cup \{\chi(v)\}$  where  $(M', L') \in \mathcal{M}$  and  $L' = L \setminus \{f(v)\}$ .

**Lemma 1.** *For any  $v \in V(G)$  and  $L \subseteq \{1, \dots, k\}$ , the batch procedure correctly computes  $\mathcal{M}_L(v)$  assuming  $\mathcal{M}_{L'}(u)$  is given for every neighbor  $u$  of  $v$  and every subset of labels  $L' \subseteq L \setminus \{f(v)\}$ .*

*Proof.* Let  $\mathcal{M}$  be the family of pairs computed by the batch procedure. Consider any pair  $(M', L') \in \mathcal{M}$  with  $L' = L \setminus \{f(v)\}$ . By construction,  $M' \subseteq M \setminus \{\chi(v)\}$  and can be written as  $M' = M'_1 \cup \dots \cup M'_\ell$ , where each  $M'_i$ ,  $1 \leq i \leq \ell$ , is a motif that has an occurrence  $V'_i$  which includes a neighbor of  $v$ . Furthermore, each  $V'_i$  is labeled by a unique set of labels  $L'_i$  such that  $L'_i \cap L'_j = \emptyset$  for all  $1 \leq j \leq \ell$ ,  $j \neq i$ . It follows that all the  $V'_i$ s are pairwise disjoint, and that  $\{v\} \cup V'_1 \cup \dots \cup V'_\ell$  is connected. Hence,  $M' \cup \{\chi(v)\}$  has an occurrence in  $G$  which is labeled by  $L' \cup \{f(v)\} = L$ , and so  $M' \cup \{\chi(v)\} \in \mathcal{M}_L(v)$ .

On the other hand, consider a motif  $M' \cup \{\chi(v)\} \in \mathcal{M}_L(v)$ . Then by definition,  $M' \cup \{\chi(v)\}$  has an occurrence  $V' \cup \{v\}$  such that the set of labels that  $f$  assigns  $V' \cup \{v\}$  is  $L$ . Let  $V'_1, \dots, V'_\ell$  be the connected components of the induced subgraph  $G[V']$ . Since  $V' \cup \{v\}$  is connected, every  $V'_i$ ,  $1 \leq i \leq \ell$ , includes a neighbor of  $v$ . Furthermore, letting  $L'_i$  denote the set of labels that  $f$  assigns  $V'_i$  for every  $1 \leq i \leq \ell$ , we have  $L' \subseteq L \setminus \{f(v)\}$  and  $L'_i \cap L'_j = \emptyset$  for all  $1 \leq i, j \leq \ell$ . It is now easy to see that the batch procedure will eventually compute the pair  $(M', L \setminus \{f(v)\})$  in its second step, and hence  $M' \cup \{\chi(v)\}$  will be added to  $\mathcal{M}_L(v)$  in its final step.  $\square$

**Lemma 2.** *Given a labeling function  $f : V(G) \rightarrow \{1, \dots, k\}$ , one can use the batch procedure iteratively in order to determine in  $\mathcal{O}(2^{5k}kn^2)$  time whether there is an occurrence of  $M$  which is distinctly labeled by  $f$ .*

*Proof.* To prove the lemma, let us first analyze the complexity of a single invocation of the batch procedure. In its first step, the batch procedure searches through at most  $2^k n$  motifs families, each consisting of at most  $2^k$  motifs. Hence, this step requires  $\mathcal{O}(2^{2k}kn)$  time. For the second step, notice that number of distinct motif and label-subset pairs is bounded by  $2^{2k}$ , and so the number of times the second step is repeated is also bounded by this term. Since each iteration of this step can be computed in  $\mathcal{O}(2^{2k}k)$  time, it follows that the second step requires  $\mathcal{O}(2^{4k}k)$  time. Accounting also for the third step, the total time of one invocation of the batch procedure is therefore  $\mathcal{O}(2^{4k}k + 2^{2k}kn) = \mathcal{O}(2^{4k}kn)$ .

It now can easily be seen that due to Lemma 1, one needs to invoke the batch procedure at most  $2^k n$  times in order to obtain  $\mathcal{M}_L(v)$  for every vertex  $v \in V(G)$  and every label subset  $L \subseteq \{1, \dots, k\}$ . It follows that in  $\mathcal{O}(2^{5k}kn^2)$  time one can obtain all necessary information to determine whether  $M$  has an occurrence which is distinctly labeled by  $f$ , and so the lemma follows.  $\square$

Note that in case  $M$  is colorful, the vertex-coloring of  $G$  distinctly colors any occurrence of  $M$ , and therefore, in this case we can determine whether  $M$  occurs in  $G$  within the time complexity given in Lemma 2. For general multiset motifs, we use the result of Alon *et al.* [2] who show how to efficiently construct a family  $\mathcal{F}$  of  $\mathcal{O}(2^{\mathcal{O}(k)} \lg n)$  functions from  $V(G)$  to  $\{1, \dots, k\}$  which is perfect. This construction builds on an earlier slightly less efficient construction of [21] and requires  $\mathcal{O}(2^{\mathcal{O}(k)} n \lg n)$  time. Using this and Lemma 2, we obtain a  $\mathcal{O}(2^{\mathcal{O}(k)} n^2 \lg n)$  algorithm for GRAPH MOTIF.

**Theorem 3.** GRAPH MOTIF can be solved in  $\mathcal{O}(2^{\mathcal{O}(k)} n^2 \lg n)$  time.

## 4 Bounded Treewidth Graphs

The treewidth parameter of graphs [22] plays a central role in designing exact algorithms for many NP-hard graph problems [3,4,5,10]. Among numerous frameworks developed over the years, we adopt the parsing mechanism developed for bounded treewidth graphs in [1]. For motifs which consist of a constant

number of colors  $c$  and graphs with treewidth smaller than some constant  $\omega$ , we present a polynomial-time algorithm with running time  $\mathcal{O}(n^{2c\omega+2c+2})$ . This should be compared with the sharp hardness result of Theorem 2. Moreover, our algorithm can also be analyzed as a fixed-parameter algorithm for parameters  $\omega$  and  $k$  which outperforms the algorithm of the previous section when the treewidth of  $G$  is sufficiently small. Due to space limitations, we only present a sketch of our result.

**Theorem 4.** *Let  $\omega$  be any positive constant. Then GRAPH MOTIF can be solved in  $\mathcal{O}(n^{2c\omega+2c+2})$  time, when  $G$  has treewidth less than  $\omega$  and  $M$  consists of  $c$  colors.*

*Proof (sketch).* The proof is sketched as follows. We employ the parsing operator point of view on bounded treewidth. In particular, we use the notion of  $\omega$ -boundaried graphs, where an  $\omega$ -boundaried graph is no more than a graph with  $\omega$  distinguished vertices, each distinctly labeled by a label in  $\{1, \dots, \omega\}$ , which are referred to as boundary vertices. The boundary vertices, together with  $\omega$ -operators, allow the construction of  $\omega$ -boundaried graphs from smaller  $\omega$ -boundaried graphs. The  $\omega$ -operators are:

1. The null operator  $\emptyset$  which creates the trivial boundaried graph with isolated vertices.
2. The binary operator  $\oplus$  that takes the disjoint union of two  $\omega$ -boundaried graphs and then identifies the  $i$ th boundary vertex of the first graph with the  $i$ th boundary vertex of the second graph.
3. The unary operator that introduces a new isolated vertex and makes this the new vertex 1 of the boundary. The old vertex 1 is removed from the boundary but not from the graph.
4. The unary operator that adds an edge between vertex 1 and vertex 2 of the boundary.
5. Unary operators that permute the labels of the boundary vertices.

A parse tree is an at-most binary rooted tree with labels corresponding to  $\omega$ -operators. The leaves are labeled with  $\emptyset$ , the internal unary nodes are labeled with unary operators, and the internal binary nodes are labeled with the binary operator  $\oplus$ . Each rooted subtree of a parse tree corresponds to an  $\omega$ -boundaried graph, where the graph at each parent is obtained by applying the corresponding operator of the parent on the  $\omega$ -boundaried graph(s) of its child(ren). We say that a parse tree *parses* an  $\omega$ -boundaried graph  $H$ , if  $H$  corresponds to the  $\omega$ -boundaried graph of the root. We extend this definition to any graph, by simply assuming that the final parsing operator removes all vertices from the boundary. Any graph of treewidth less than  $\omega$  can be parsed by a parse tree with  $\mathcal{O}(\omega n)$  nodes [1].

Define a *checklist item* for a  $w$ -boundaried graph to consist of the following information: (1) A partition  $\pi$  of the set of boundary vertices. Let  $X$  denote the set of boundary vertices, and write  $\pi = (X_1, \dots, X_r)$  where  $r \leq \omega$ , and the  $X_i$  denote the sets of the partition  $\pi$ . (2) A *motif family*  $\mathcal{M}_\pi = (M_0, M_1, \dots, M_r)$  of

length  $r + 1$ , where each  $M_i$  is non-empty except maybe  $M_0$  and  $M_i \subseteq M$ . (Note that the number of distinct checklist items is at most  $\omega^\omega (n^c)^{\omega+1} = w^\omega n^{c\omega+c}$ , where a better bound is given by replacing  $w^\omega$  with  $Bell(\omega)$ , the number of distinct partitions of an  $\omega$ -element set.)

We say that a *checklist item*  $\alpha$  as above is *positive* for a  $\omega$ -boundaried graph  $A$  if there is a set of  $r + 1$  vertex-disjoint subsets  $V_0, \dots, V_r \subseteq V(A)$  satisfying the following conditions:

1.  $V_0 \cap X = \emptyset$ .
2. For  $i = 1, \dots, r$ ,  $V_i \cap X = X_i$ .
3. For  $i = 0, \dots, r$ ,  $V_i$  is an occurrence of  $M_i$  in  $A$

Define the *inventory*  $inv(A)$  of the  $w$ -boundaried graph  $A$  to be the set of all checklist items that are positive for  $A$ .

*Claim.* Whether a motif occurs in a  $\omega$ -boundaried graph  $A$  can be determined from  $inv(A)$  in time linear in the size of the inventory.

Our algorithm proceeds as follows. It first computes a parse tree of  $G$ , and then computes, from the leaves up to the root, the inventories of the  $\omega$ -boundaried graphs corresponding to the nodes of the  $\omega$  parse tree. Let  $A$  be the trivial boundary graph obtained by the null operator  $\emptyset$ . In this base case,  $inv(A)$  consists of single checklist item with a partition  $\pi = (X_1, \dots, X_r)$  that partitions the boundary vertices into singletons, and motif family  $\mathcal{M}_\pi = (\emptyset, M_1, \dots, M_r)$  where  $M_i$  consists of the color of the single boundary vertex in  $X_i \in \pi$ , for all  $i = 1, \dots, r$ . For boundary graphs obtained by unary operations, computing the inventory is almost equally easy.

*Claim.* One can compute  $inv(op(A))$  from  $inv(A)$ .

We proceed to describe the computation for the  $\oplus$  operator. Let  $A$  and  $B$  be two boundaried graphs over the same boundary vertex set  $X$ . If  $\alpha \in inv(A)$  and  $\beta \in inv(B)$  then we define the checklist item  $\alpha \oplus \beta$  as follows. As per the definition of a checklist item, we must give two pieces of information to describe  $\alpha \oplus \beta$ : (1) a partition  $\pi_{\alpha \oplus \beta}$  of  $X$ , and (2) a motif family  $\mathcal{M}_{\pi_{\alpha \oplus \beta}}$  for this partition. Let  $\pi_\alpha$  ( $\pi_\beta$ ) denote the partition of  $X$  for the checklist item  $\alpha$  ( $\beta$ ). Let  $\equiv_\alpha$  ( $\equiv_\beta$ ) be the equivalence relation on  $X$  defined by  $\pi_\alpha$  ( $\pi_\beta$ ). The partition  $\pi_{\alpha \oplus \beta}$  corresponds to the reflexive and transitive closure of the relation  $\equiv_\alpha \cup \equiv_\beta$ . The motif family  $\mathcal{M}_{\pi_{\alpha \oplus \beta}}$  is obtained by adding and subtracting colors of the motifs of  $\mathcal{M}_{\pi_\alpha}$  and  $\mathcal{M}_{\pi_\beta}$  in the natural way.

*Claim.* We can compute  $inv(A \oplus B)$  as  $\{\alpha, \beta, \alpha \oplus \beta \mid \alpha \in inv(A), \beta \in inv(B)\}$ .

Hence given a parse tree of  $G$ , we have to perform at most  $\omega n$  such “multiplications of inventories”. Since each inventory has size bounded by  $\omega^\omega n^{c\omega+c}$ , and since a single multiplication between two inventories requires  $\mathcal{O}(n)$  time, this gives a running time of  $\mathcal{O}(n^{2c\omega+2c+2})$  for all inventory multiplications. Since the computation on the  $\oplus$  operator requires more time than the computation on any other operator, the entire algorithm runs within this time bound. □

Showing that our algorithm is a fixed-parameter algorithm for parameters  $\omega$  and  $k$  involves pretty much the same analysis. The only difference is that here we bound the total number of distinct checklist items by  $\omega^\omega (2^k)^{\omega+1} = \omega^{\omega} 2^{k\omega+k}$ .

## 5 On Trees and Motifs with Bounded Number of Colors

Although Theorem 4 gives a nice complementary result to the sharp hardness result of Theorem 2, it still leaves a certain gap. In the following section we aim to close this gap, by proving that GRAPH MOTIF, parameterized by the number of colors  $c$  in  $M$ , is W[1]-hard for trees. Essentially, this means that unless unlikely collapses occur in parameterized complexity theory, there is no fixed-parameter algorithm for parameter  $c$ , even in the restricted case of trees. We refer readers unfamiliar with the concept of parameterized reductions to [12].

**Theorem 5.** *The GRAPH MOTIF problem, parameterized by the number of colors  $c$  in the motif, is W[1]-hard for trees.*

*Proof.* We present a reduction from CLIQUE which is known to be W[1]-hard [12]. Recall that in CLIQUE we are given a simple graph  $H$  and an integer  $\kappa$ , the parameter, and we are asked to determine whether  $H$  has a subset of  $\kappa$  vertices which are pairwise adjacent. Given an instance  $\langle H, \kappa \rangle$  of CLIQUE, we describe a rooted tree  $G = T$  colored with  $1 + \kappa + 2\kappa(\kappa - 1) + \binom{\kappa}{2}$  colors. We let  $m$  denote the number of edges of  $H$ , i.e.  $m = |E(H)|$ .

- The root of  $T$  is colored  $a$ .
- The root has  $\kappa \cdot |V(H)|$  children organized into  $\kappa$  groups  $S(1) \dots S(\kappa)$ . The group of  $|V(H)|$  children  $S(i)$  consists of the nodes  $s(i, u)$ , where  $u \in V(H)$ . The color of each node in  $S(i)$  is  $b(i)$ .
- From each node  $s(i, u)$  hang  $\kappa - 1$  groups of paths. The groups are  $P(i, u, j)$  for every  $j \in \{1 \dots, \kappa\} \setminus \{i\}$ . There is one path  $p(i, u, j, v) \in P(i, u, j)$  for each edge  $\{u, v\} \in E(H)$  that is incident to  $u$  in  $H$ .

The path  $p(i, u, j, v)$  begins with a vertex colored  $c(i, j)$  and ends with a vertex colored  $d(i, j)$ , and otherwise consists of some number  $m(i, u, j, v)$  of internal vertices colored by  $e(i, j) = e(j, i)$ . There is an important detail to note here. If  $i < j$ , then  $c(i, j)$  and  $c(j, i)$  are different colors, whereas  $e(i, j)$  and  $e(j, i)$  are the same color. We call the colors  $e(i, j)$  the *edge colors*. The number  $m(i, u, j, v)$  is calculated as follows. Number the edges in  $E(H)$  from 1 to  $m$ , letting  $l(uv)$  denote the number assigned to the edge  $\{u, v\} \in E(H)$ . We define:

$$m(i, u, j, v) = \begin{cases} l(uv) & i < j \\ 3m - l(uv) & i > j. \end{cases}$$

The motif  $M$  consists of one of each of every color other than the edge colors, and  $3m$  elements colored by each edge color. Thus,  $M$  consists of  $c = 1 + \kappa + 2\kappa(\kappa - 1) + \binom{\kappa}{2}$  different colors, and  $|M| = 1 + \kappa + 2\kappa(\kappa - 1) + 3m\binom{\kappa}{2}$ . This completes the construction of our instance  $\langle (G, M), c \rangle$  for GRAPH MOTIF.

We claim that  $H$  has a subset of  $\kappa$  pairwise adjacent vertices if and only if  $T$  has a subtree  $T'$  which is an occurrence of  $M$ . Suppose that the vertices  $v_1, \dots, v_\kappa$  are pairwise adjacent in  $H$ . The subtree  $T'$  consists of:

- The root which is colored  $a$ .
- The  $\kappa$  children of the root  $s(i, v_i)$  for all  $1 \leq i \leq \kappa$ , where  $s(i, v_i)$  is colored  $b(i)$ .
- The  $\kappa(\kappa - 1)$  paths  $p(i, v_i, j, v_j)$ . The path  $p(i, v_i, j, v_j)$  begins with a node colored  $c(i, j)$  and ends with a node colored  $d(i, j)$  for all  $1 \leq i, j \leq \kappa$ ,  $i \neq j$ . Note that the path  $p(i, v_i, j, v_j)$  is pendant from  $s(i, v_i)$  since  $v_i$  and  $v_j$  are adjacent in  $H$ . Together, the two complementary paths  $p(i, v_i, j, v_j)$  and  $p(j, v_j, i, v_i)$  contain  $3m$  nodes colored  $e(i, j)$ .

In the other direction, suppose that the subtree  $T'$  of  $T$  is an occurrence of  $M$ . Then  $T'$  must include the root of  $T$ , since it is the only node colored  $a$ . Furthermore,  $T'$  must contain exactly one node in each of the groups  $S(i)$ ,  $1 \leq i \leq \kappa$ , since nodes in each  $S(i)$  are all colored  $b(i)$ . Suppose these nodes are  $s(1, v_1), \dots, s(\kappa, v_\kappa)$ . We argue that the vertices  $v_1, \dots, v_\kappa$  are pairwise adjacent in  $H$ .

In order for  $T'$  to be an occurrence of  $M$  in  $T$ ,  $T'$  must contain exactly one pendant path in each of the groups of paths  $P(i, v_i, j)$  for any  $1 \leq i, j \leq \kappa$ ,  $i \neq j$ , and nothing further. To see this, note that  $T'$  must contain at least one path in each of the groups of paths  $P(i, v_i, j)$  in order for  $T'$  to contain a node colored  $d(i, j)$ . But containing one such path prevents  $T'$  from including any nodes of other paths in  $P(i, v_i, j)$ , else  $T'$  would contain too many nodes of color  $c(i, j)$ .

It follows that for any pair of indices  $i, j$  with  $1 \leq i < j \leq \kappa$ ,  $T'$  includes exactly two paths  $p(i, v_i, j, x)$  and  $p(j, v_j, i, y)$  that contain nodes of color  $e(i, j) = e(j, i)$ . Since  $M$  contains exactly  $3m$  elements colored by  $e(i, j)$ , it follows that  $x = v_j$  and  $y = v_i$ , since  $p(i, v_i, j, v_j)$  and  $p(j, v_j, i, v_i)$  are the only two paths in  $T$  with nodes colored  $e(i, j)$  that together have exactly  $3m$  nodes of this color. But then, by construction of  $T$ ,  $v_i$  and  $v_j$  must be adjacent in  $H$ .  $\square$

## References

1. Abrahamson, K.R., Fellows, M.R.: Finite automata, bounded treewidth, and well-quasiordering. In: N. Robertson and P. Seymour (eds.) Graph Structure Theory, pp. 539–564 (1993)
2. Alon, N., Yuster, R., Zwick, U.: Color coding. Journal of the ACM 42(4), 844–856 (1995)
3. Arnborg, S.: Efficient algorithms for combinatorial problems on graphs with bounded decomposability. A survey. BIT Numerical Mathematics 25(1), 2–23 (1985)
4. Arnborg, S., Proskurowski, A.: Linear time algorithms for NP-hard problems restricted to partial  $k$ -trees. Discrete Applied Mathematics 23, 11–24 (1989)
5. Bodlaender, H.L.: A tourist guide through treewidth. Acta Cybernetica 11, 1–23 (1993)

6. Bodlaender, H.L., Fluitier, L.E.: Intervalizing  $k$ -colored graphs. In: Fülöp, Z., Gecseg, F. (eds.) Automata, Languages and Programming. LNCS, vol. 944, pp. 87–98. Springer, Heidelberg (1995)
7. Bodlaender, H.L., Fellows, M.R., Langston, M.A., Ragan, M.A., Rosamond, F.A., Weyer, M.: Kernelization for convex recoloring. In: Proceedings of the 2nd workshop on Algorithms and Complexity in Durham (ACiD), pp. 23–35 (2006)
8. Bodlaender, H.L., Fellows, M.R., Warnow, T.: Two strikes against perfect phylogeny. In: Kuich, W. (ed.) Automata, Languages and Programming. LNCS, vol. 623, pp. 273–283. Springer, Heidelberg (1992)
9. Chor, B., Fellows, M.R., Ragan, M.A., Rosamond, F.A., Snir, S.: Connected coloring completion for general graphs: Algorithms and complexity – Manuscript (2007)
10. Cornil, D.G., Keil, J.M.: A dynamic programming approach to the dominating set problem on  $k$ -trees. *SIAM Journal on Algebraic and Discrete Methods* 8(4), 535–543 (1987)
11. Deville, Y., Gilbert, D., Helden, J.V., Wodak, S.J.: An overview of data models for the analysis of biochemical pathways. *Briefings in Bioinformatics* 4(3), 246–259 (2003)
12. Downey, R., Fellows, M.: *Parameterized Complexity*. Springer, Heidelberg (1999)
13. Fellows, M.R., Hallett, M.T., Wareham, H.T.: DNA physical mapping: Three ways difficult. In: Lengauer, T. (ed.) *ESA 1993*. LNCS, vol. 726, pp. 157–168. Springer, Heidelberg (1993)
14. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York (1979)
15. Golubic, M., Kaplan, H., Shamir, R.: On the complexity of DNA physical mapping. *Advances in Applied Mathematics* 15, 251–261 (1994)
16. Ideker, T., Karp, R.M., Scott, J., Sharan, R.: Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology* 13(2), 133–144 (2006)
17. Kelley, B.P., Sharan, R., Karp, R.M., Sittler, T., Root, D.E., Stockwell, B.R., Ideker, T.: Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proceedings of the National Academy of Sciences of the United States of America* 100(20), 11394–11399 (2003)
18. Lacroix, V., Fernandes, C.G., Sagot, M.-F.: Motif search in graphs: Application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 3(4), 360–368 (2006)
19. McMorris, F.R., Warnow, T.J., Wimer, T.: Triangulating vertex-colored graphs. *SIAM Journal on Discrete Mathematics* 7(2), 296–306 (1994)
20. Moran, S., Snir, S.: Convex recolorings of strings and trees: Definitions, hardness results and algorithms. In: Dehne, F., López-Ortiz, A., Sack, J.-R. (eds.) *WADS 2005*. LNCS, vol. 3608, pp. 218–232. Springer, Heidelberg (2005)
21. Naor, J., Naor, M.: Small-bias probability spaces: Efficient constructions and applications. In: Proceedings of the 25th annual ACM Symposium on Theory Of Computing (STOC), pp. 213–223. ACM Press, New York (1990)
22. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. *SIAM Journal of Algorithms* 7, 309–322 (1986)



# Parameterized Algorithms for Directed Maximum Leaf Problems

Noga Alon<sup>1</sup>, Fedor V. Fomin<sup>2</sup>, Gregory Gutin<sup>3</sup>, Michael Krivelevich<sup>1</sup>,  
and Saket Saurabh<sup>2,4</sup>

<sup>1</sup> Department of Mathematics, Tel Aviv University  
Tel Aviv 69978, Israel

{nogaa,krivelev}@post.tau.ac.il

<sup>2</sup> Department of Informatics, University of Bergen  
POB 7803, 5020 Bergen, Norway

{fedor.fomin,saket}@ii.uib.no

<sup>3</sup> Department of Computer Science  
Royal Holloway, University of London  
Egham, Surrey TW20 0EX, UK

gutin@cs.rhul.ac.uk

<sup>4</sup> The Institute of Mathematical Sciences  
Chennai, 600 017, India

saket@imsc.res.in

**Abstract.** We prove that finding a rooted subtree with at least  $k$  leaves in a digraph is a fixed parameter tractable problem. A similar result holds for finding rooted spanning trees with many leaves in digraphs from a wide family  $\mathcal{L}$  that includes all strong and acyclic digraphs. This settles completely an open question of Fellows and solves another one for digraphs in  $\mathcal{L}$ . Our algorithms are based on the following combinatorial result which can be viewed as a generalization of many results for a ‘spanning tree with many leaves’ in the undirected case, and which is interesting on its own: If a digraph  $D \in \mathcal{L}$  of order  $n$  with minimum in-degree at least 3 contains a rooted spanning tree, then  $D$  contains one with at least  $(n/2)^{1/5} - 1$  leaves.

## 1 Introduction

The MAXIMUM LEAF SPANNING TREE problem (finding a spanning tree with the maximum number of leaves in a connected undirected graph) is an intensively studied problem from an algorithmic as well as a combinatorial point of view [5,7,10,13,17,22,30]. It fits into the broader class of spanning tree problems on which hundreds of papers have been written; see e.g. the book by Wu and Chao [34]. It is known to be NP-hard [18], and APX-hard [16], but can be approximated efficiently with multiplicative factor 3 [26] and even 2 [30].

In this paper, we initiate the combinatorial and algorithmic study of two natural generalizations of the problem to digraphs. We say that a subgraph  $T$  of a digraph  $D$  is an *out-tree* if  $T$  is an oriented tree with only one vertex  $s$  of in-degree zero (called *the root*). The vertices of  $T$  of out-degree zero are called *leaves*.

If  $T$  is a spanning out-tree, i.e.  $V(T) = V(D)$ , then  $T$  is called an *out-branching* of  $D$ . Given a digraph  $D$ , the DIRECTED MAXIMUM LEAF OUT-BRANCHING problem is the problem of finding an out-branching in  $D$  with the maximum possible number of leaves. Denote this maximum by  $\ell_s(D)$ . When  $D$  has no out-branching, we write  $\ell_s(D) = 0$ . Similarly, the DIRECTED MAXIMUM LEAF OUT-TREE problem is the problem of finding an out-tree in  $D$  with the maximum possible number of leaves, which we denote by  $\ell(D)$ . Both these problems are equivalent for connected undirected graphs, as any maximum leaf tree can be extended to a maximum leaf spanning tree with the same number of leaves.

Notice that  $\ell(D) \geq \ell_s(D)$  for each digraph  $D$ . Let  $\mathcal{L}$  be the family of digraphs  $D$  for which either  $\ell_s(D) = 0$  or  $\ell_s(D) = \ell(D)$ . It is easy to see that  $\mathcal{L}$  contains all strong and acyclic digraphs.

We investigate the above two problems from the *parameterized complexity* point of view. Parameterized Complexity is a recent approach to deal with intractable computational problems having some parameters that can be relatively small with respect to the input size. This area has been developed extensively during the last decade. For decision problems with input size  $n$ , and a parameter  $k$ , the goal is to design an algorithm with runtime  $f(k)n^{O(1)}$  where  $f$  is a function of  $k$  alone. Problems having such an algorithm are said to be fixed parameter tractable (FPT). The book by Downey and Fellows [11] provides a good introduction to the topic of parameterized complexity. For recent developments see the books by Flum and Grohe [15] and by Niedermeier [28].

The parameterized version of the DIRECTED MAXIMUM LEAF OUT-BRANCHING (the DIRECTED MAXIMUM LEAF OUT-TREE) problem is defined as follows: Given a digraph  $D$  and a positive integral parameter  $k$ , is  $\ell_s(D) \geq k$  ( $\ell(D) \geq k$ )? We denote the parameterized versions of the DIRECTED MAXIMUM LEAF OUT-BRANCHING and the DIRECTED MAXIMUM LEAF OUT-TREE problems by  $k$ -DMLOB and  $k$ -DMLOT respectively.

While the parameterized complexity of almost all natural problems on undirected graphs is well understood, the world of digraphs is still wide open. The main reason for this anomaly is that most of the techniques developed for undirected graphs cannot be used or extended to digraphs. One of the most prominent examples is the FEEDBACK VERTEX SET problem, which is easily proved to be FPT for undirected graphs, while its parameterized complexity on digraphs is a long standing open problem in the area. In what follows we briefly explain why the standard techniques for the MAXIMUM LEAF SPANNING TREE problem on undirected graphs cannot be used for its generalizations to digraphs.

- The Graph Minors Theory of Robertson and Seymour [31] is a powerful (yet non-constructive) technique for establishing membership in FPT. For example, this machinery can be used to show that the MAXIMUM LEAF SPANNING TREE problem is FPT for undirected graphs (see [12]). However, Graph Minors Theory for digraphs is still in a preliminary stage and at the moment cannot be used as a tool for tackling interesting directed graph problems.

- Bodlaender [3] used the following arguments to prove that the MAXIMUM LEAF SPANNING TREE problem is FPT: If an undirected graph  $G$  contains a star  $K_{1,k}$  as a minor, then it is possible to construct a spanning tree with at least  $k$  leaves from this minor. Otherwise, there is no  $K_{1,k}$  minor in  $G$ , and it is possible to prove that the treewidth of  $G$  is at most  $f(k)$ . Thus, dynamic programming can be used to decide whether there is a tree with  $k$  leaves. This approach does not work on directed graphs because containing a big out-tree as a minor does not imply the existence of an out-branching or out-tree with many leaves in the original graph. In short, the properties of having no out-branching with at least  $k$  leaves or having no out-tree with  $k$  leaves are not minor closed.
- The seemingly most efficient approach for designing FPT algorithms for undirected graphs is based on a combination of combinatorial bounds and preprocessing rules for handling vertices of small degrees. Kleitman and West [22] and Linial and Sturtevant [25] showed that every connected undirected graph  $G$  on  $n$  vertices with minimum degree at least 3 has a spanning tree with at least  $n/4 + 2$  leaves. Bonsma et al. [5] combined this combinatorial result with clever preprocessing rules to obtain the fastest known algorithm for the  $k$ -MAXIMUM LEAF SPANNING TREE problem, running in time  $O(n^3 + 9.4815^k k^3)$ . It is not clear how to devise a similar approach for digraphs.

**Our Contribution.** We obtain a number of combinatorial and algorithmic results for the DIRECTED MAXIMUM LEAF OUT-BRANCHING and the DIRECTED MAXIMUM LEAF OUT-TREE problems. Our main combinatorial result (Theorem 1) is the proof that for every digraph  $D \in \mathcal{L}$  of order  $n$  with minimum in-degree at least 3,  $\ell_s(D) \geq (n/2)^{1/5} - 1$  provided  $\ell_s(D) > 0$ . This can be viewed as a generalization of many combinatorial results for undirected graphs related to the existence of spanning trees with many leaves [19,22,25].

Our main algorithmic contributions are fixed parameter tractable algorithms for the  $k$ -DMLOB and the  $k$ -DMLOT problems for digraphs in  $\mathcal{L}$  and for all digraphs, respectively. The algorithms are based on a decomposition theorem which uses ideas from the proof of the main combinatorial result. More precisely, we show that either a digraph contains a structure that can be extended to an out-branching with many leaves, or the pathwidth of the underlying undirected graph is small. This settles completely an open question of Mike Fellows [6,14,21] and solves another one for digraphs in  $\mathcal{L}$ .

## 2 Preliminaries

Let  $D$  be a digraph. By  $V(D)$  and  $A(D)$  we represent the vertex set and arc set of  $D$ , respectively. An *oriented graph* is a digraph with no directed 2-cycle. Given a subset  $V' \subseteq V(D)$  of a digraph  $D$ , let  $D[V']$  denote the subgraph induced on  $V'$ . The *underlying undirected graph*  $UN(D)$  of  $D$  is obtained from  $D$  by omitting all orientations of arcs and by deleting one edge from each resulting pair of parallel edges. The *connectivity components* of  $D$  are the subgraphs of  $D$  induced by the

vertices of connected components of  $UN(D)$ . A vertex  $y$  of  $D$  is an *in-neighbor* (*out-neighbor*) of a vertex  $x$  if  $yx \in A$  ( $xy \in A$ ). The *in-degree*  $d^-(x)$  (*out-degree*  $d^+(x)$ ) of a vertex  $x$  is the number of its in-neighbors (out-neighbors). A vertex  $s$  of a digraph  $D$  is a *source* if the in-degree of  $s$  is 0.

A digraph  $D$  is *strong* if there is a directed path from every vertex of  $D$  to every other vertex of  $D$ . A *strong component* of a digraph  $D$  is a maximal strong subgraph of  $D$ . A strong component  $S$  of a digraph  $D$  is a *source strong component* if no vertex of  $S$  has an in-neighbor in  $V(D) \setminus V(S)$ . The following simple result gives necessary and sufficient conditions for a digraph to have an out-branching.

**Proposition 1** ([2]). *A digraph  $D$  has an out-branching if and only if  $D$  has a unique source strong component.*

This assertion allows us to check whether  $\ell_s(D) > 0$  in time  $O(|V(D)| + |A(D)|)$ . Thus, we will often assume, in the rest of the paper, that the digraph  $D$  under consideration has an out-branching.

Let  $P = u_1u_2 \dots u_q$  be a directed path in a digraph  $D$ . An arc  $u_iu_j$  of  $D$  is a *forward* (*backward*) *arc for  $P$*  if  $i \leq j - 2$  ( $j < i$ , respectively). Every backward arc of the type  $v_{i+1}v_i$  is called *double*.

For a natural number  $n$ ,  $[n]$  denotes the set  $\{1, 2, \dots, n\}$ .

The notions of treewidth and pathwidth were introduced by Robertson and Seymour in [32] and [33] (see [3] and [27] for surveys).

A *tree decomposition* of an (undirected) graph  $G$  is a pair  $(X, U)$  where  $U$  is a tree whose vertices we will call *nodes* and  $X = (\{X_i \mid i \in V(U)\})$  is a collection of subsets of  $V(G)$  such that

1.  $\bigcup_{i \in V(U)} X_i = V(G)$ ,
2. for each edge  $\{v, w\} \in E(G)$ , there is an  $i \in V(U)$  such that  $v, w \in X_i$ , and
3. for each  $v \in V(G)$  the set of nodes  $\{i \mid v \in X_i\}$  forms a subtree of  $U$ .

The *width* of a tree decomposition  $(\{X_i \mid i \in V(U)\}, U)$  equals  $\max_{i \in V(U)} \{|X_i| - 1\}$ . The *treewidth* of a graph  $G$  is the minimum width over all tree decompositions of  $G$ .

If in the definitions of a tree decomposition and treewidth we restrict  $U$  to be a tree with all vertices of degree at most 2 (i.e., a path) then we have the definitions of path decomposition and pathwidth. We use the notation  $tw(G)$  and  $pw(G)$  to denote the treewidth and the pathwidth of a graph  $G$ .

We also need an equivalent definition of pathwidth in terms of vertex separators with respect to a linear ordering of the vertices. Let  $G$  be a graph and let  $\sigma = (v_1, v_2, \dots, v_n)$  be an ordering of  $V(G)$ . For  $j \in [n]$  put  $V_j = \{v_i : i \in [j]\}$  and denote by  $\partial V_j$  all vertices of  $V_j$  that have neighbors in  $V \setminus V_j$ . Setting

$$vs(G, \sigma) = \max_{i \in [n]} |\partial V_i|,$$

we define the *vertex separation* of  $G$  as

$$vs(G) = \min\{vs(G, \sigma) : \sigma \text{ is an ordering of } V(G)\}.$$

The following assertion is well-known. It follows directly from the results of Kirousis and Papadimitriou [24] on interval width of a graph, see also [23].

**Proposition 2** ([23,24]). *For any graph  $G$ ,  $vs(G) = pw(G)$ .*

### 3 Combinatorial Lower Bounds on $\ell(D)$ and $\ell_s(D)$

Let  $\mathcal{D}$  be a family of digraphs. Notice that if we can show that  $\ell_s(D) \geq g(n)$  for every digraph  $D \in \mathcal{D}$  of order  $n$ , where  $g(n)$  is tending to infinity as  $n$  tends to infinity, then  $k$ -DMLOB is FPT on  $\mathcal{D}$ . Indeed,  $g(n) < k$  holds only for digraphs with less than some  $G(k)$  vertices and we can generate all out-branchings in such a digraph in time bounded by a function of  $k$ .

Unfortunately, bounds of the type  $\ell_s(D) \geq g(n)$  are not valid for all strong digraphs. Nevertheless, such bounds hold for wide classes of digraphs as we show in the rest of this section.

The following assertion shows that  $\mathcal{L}$  includes a large number of digraphs including all strong and acyclic digraphs (and, also, well-studied classes of semi-complete multipartite digraphs and quasi-transitive digraphs, see [2] for the definitions).

**Proposition 3.** *Suppose that a digraph  $D$  satisfies the following property: for every pair  $R$  and  $Q$  of distinct strong components of  $D$ , if there is an arc from  $R$  to  $Q$  then each vertex of  $Q$  has an in-neighbor in  $R$ . Then  $D \in \mathcal{L}$ .*

*Proof.* Let  $T$  be a maximal out-tree of  $D$  with  $\ell(D)$  leaves. We may assume that  $\ell_s(D) > 0$  and  $V(T) \neq V(D)$ . Let  $H$  be the unique source strong component of  $D$  and let  $r$  be the root of  $T$ . Observe that  $r \in V(H)$  as otherwise we could extend  $T$  by adding to it an arc  $ur$ , where  $u$  is some vertex outside the strong component containing  $r$ . Let  $C$  be a strong component containing a vertex from  $T$ . Observe that  $V(C) \cap V(T) = V(C)$  as otherwise we could extend  $T$  by appending to it some arc  $uv$ , where  $u \in V(C) \cap V(T)$  and  $v \in V(C) \setminus V(T)$ . Similarly, one can see that  $T$  must contain vertices from all strong components of  $D$ . Thus,  $V(T) = V(D)$ , a contradiction. □

#### 3.1 Digraphs with Restricted In-Degree

**Lemma 1.** *Let  $D$  be an oriented graph of order  $n$  with every vertex of in-degree 2 and let  $D$  have an out-branching. If  $D$  has no out-tree with  $k$  leaves, then  $n \leq 2k^5$ .*

*Proof.* Assume that  $D$  has no out-tree with  $k$  leaves. Consider an out-branching  $T$  of  $D$  with  $p$  leaves (clearly  $p < k$ ). Start from the empty collection  $\mathcal{P}$  of vertex-disjoint directed paths. Choose a directed path  $R$  from the root of  $T$  to a leaf, add  $R$  to  $\mathcal{P}$  and delete  $V(R)$  from  $T$ . Repeat this for each of the out-trees comprising  $T - V(R)$ . By induction on the number of leaves, it is easy to see that this process provides a collection  $\mathcal{P}$  of  $p$  vertex-disjoint directed paths covering all vertices of  $D$ .

Let  $P \in \mathcal{P}$  have  $q \geq n/p$  vertices and let  $P' \in \mathcal{P} \setminus \{P\}$ . There are at most  $k - 1$  vertices on  $P$  with in-neighbors on  $P'$  since otherwise we could choose a set  $X$  of at least  $k$  vertices on  $P$  for which there were in-neighbors on  $P'$ . The vertices of  $X$  would be leaves of an out-tree formed by the vertices  $V(P') \cup X$ . Thus, there are  $m \leq (k - 1)(p - 1) \leq (k - 1)(k - 2)$  vertices of  $P$  with in-neighbors outside  $P$  and at least  $q - (k - 2)(k - 1)$  vertices of  $P$  have both in-neighbors on  $P$ .

Let  $P = u_1 u_2 \dots u_q$ . Suppose that there are  $2(k - 1)$  indices

$$i_1 < j_1 \leq i_2 < j_2 \leq \dots \leq i_{k-1} < j_{k-1}$$

such that each  $u_{i_s} u_{j_s}$  is a forward arc for  $P$ . Then the arcs

$$\{u_{i_s} u_{j_s}, u_{j_s} u_{j_s+1}, \dots, u_{i_{s+1}-1} u_{i_{s+1}} : 1 \leq s \leq k - 2\} \cup \{u_{i_{k-1}} u_{j_{k-1}}\} \cup \{u_{i_s} u_{i_{s+1}} : 1 \leq s \leq k - 1\}$$

form an out-tree with  $k$  leaves, a contradiction.

Let  $f$  be the number of forward arcs for  $P$ . Consider the graph  $G$  whose vertices are all the forward arcs and a pair  $u_i u_j, u_s u_r$  of forward arcs are adjacent in  $G$  if the intervals  $[i, j - 1]$  and  $[s, r - 1]$  of the real line intersect. Observe that  $G$  is an interval graph and, thus, a perfect graph. By the result of the previous paragraph, the independence number of  $G$  is less than  $k - 1$ . Thus, the chromatic number of  $G$  and the order  $g$  of its largest clique  $Q$  is at least  $f/(k - 2)$ . Let  $V(Q) = \{u_{i_s} u_{j_s} : 1 \leq s \leq g\}$  and let  $h = \min\{j_s - 1 : 1 \leq s \leq g\}$ . Observe that each interval  $[i_s, j_s - 1]$  contains  $h$ . Therefore, we can form an out-tree with vertices

$$\{u_1, u_2, \dots, u_h\} \cup \{u_{j_s} : 1 \leq s \leq g\}$$

in which  $\{u_{j_s} : 1 \leq s \leq g\}$  are leaves. Hence we have  $\frac{f}{k-2} \leq k - 1$  and, thus,  $f \leq (k - 2)(k - 1)$ .

Let  $uv$  be an arc of  $A(D) \setminus A(P)$  such that  $v \in V(P)$ . There are three possibilities: (i)  $u \notin V(P)$ , (ii)  $u \in V(P)$  and  $uv$  is forward for  $P$ , (iii)  $u \in V(P)$  and  $uv$  is backward for  $P$ . By the inequalities above for  $m$  and  $f$ , we conclude that there are at most  $2(k - 2)(k - 1)$  vertices on  $P$  which are not terminal vertices (i.e., heads) of backward arcs. Consider a path  $R = v_0 v_1 \dots v_r$  formed by backward arcs. Observe that the arcs  $\{v_i v_{i+1} : 0 \leq i \leq r - 1\} \cup \{v_j v_j^+ : 1 \leq j \leq r\}$  form an out-tree with  $r$  leaves, where  $v_j^+$  is the out-neighbor of  $v_j$  on  $P$ . Thus, there is no path of backward arcs of length more than  $k - 1$ .

If the in-degree of  $u_1$  in  $D[V(P)]$  is 2, remove one of the backward arcs terminating at  $u_1$ . Observe that now the backward arcs for  $P$  form a vertex-disjoint collection of out-trees with roots at vertices that are not terminal vertices of backward arcs. Therefore, the number of the out-trees in the collection is at most  $2(k - 2)(k - 1)$ . Observe that each out-tree in the collection has at most  $k - 1$  leaves and thus its arcs can be decomposed into at most  $k - 1$  paths, each of length at most  $k$ . Hence, the original total number of backward arcs for  $P$  is at most  $2k(k - 2)(k - 1)^2 + 1$ . On the other hand, it is at least  $(q - 1) - 2(k - 2)(k - 1)$ . Thus,  $(q - 1) - 2(k - 2)(k - 1) \leq 2k(k - 2)(k - 1)^2 + 1$ . Combining this inequality with  $q \geq n/(k - 1)$ , we conclude that  $n \leq 2k^5$ . □

**Theorem 1.** *Let  $D$  be a digraph in  $\mathcal{L}$  with  $\ell_s(D) > 0$ .*

- (a) *If  $D$  is an oriented graph with minimum in-degree at least 2, then  $\ell_s(D) \geq (n/2)^{1/5} - 1$ .*
- (b) *If  $D$  is a digraph with minimum in-degree at least 3, then  $\ell_s(D) \geq (n/2)^{1/5} - 1$ .*

*Proof.* (a) Let  $T$  be an out-branching of  $D$ . Delete some arcs from  $A(D) \setminus A(T)$ , if needed, such that the in-degree of each vertex of  $D$  becomes 2. Now the inequality  $\ell_s(D) \geq (n/2)^{1/5} - 1$  follows from Lemma 1 and the definition of  $\mathcal{L}$ .

(b) Let  $T$  be an out-branching of  $D$ . Let  $P$  be the path formed in the proof of Lemma 1. (Note that  $A(P) \subseteq A(T)$ .) Delete every double arc of  $P$ , in case there are any, and delete some more arcs from  $A(D) \setminus A(T)$ , if needed, to ensure that the in-degree of each vertex of  $D$  becomes 2. It is not difficult to see that the proof of Lemma 1 remains valid for the new digraph  $D$ . Now the inequality  $\ell_s(D) \geq (n/2)^{1/5} - 1$  follows from Lemma 1 and the definition of  $\mathcal{L}$ . □

It is not difficult to give examples showing that the restrictions on the minimum in-degrees in Theorem 1 are optimal. Indeed, any directed cycle  $C$  is a strong oriented graph with all in-degrees 1 for which  $\ell_s(C) = 1$  and any directed double cycle  $D$  is a strong digraph with in-degrees 2 for which  $\ell_s(D) = 2$  (a *directed double cycle* is a digraph obtained from an undirected cycle by replacing every edge  $xy$  with two arcs  $xy$  and  $yx$ ).

## 4 Parameterized Algorithms for $k$ -DMLOB and $k$ -DMLOT

In the previous section, we gave lower bounds on  $\ell(D)$  and  $\ell_s(D)$  for digraphs  $D \in \mathcal{L}$  with minimum in-degree at least 3. These bounds trivially imply the fixed parameter tractability of the  $k$ -DMLOB and the  $k$ -DMLOT problems for this class of digraphs. Here we extend these FPT results to digraphs in  $\mathcal{L}$  for  $k$ -DMLOB and to all digraphs for  $k$ -DMLOT. We prove a decomposition theorem which either outputs an out-tree with  $k$  leaves or provides a path decomposition of the underlying undirected graph of width  $O(k^2)$  in polynomial time.

**Theorem 2.** *Let  $D$  be a digraph in  $\mathcal{L}$  with  $\ell_s(D) > 0$ . Then either  $\ell_s(D) \geq k$  or the underlying undirected graph of  $D$  is of pathwidth at most  $2k^2$ .*

*Proof.* Let  $D$  be a digraph in  $\mathcal{L}$  with  $0 < \ell_s(D) < k$ . Let us choose an out-branching  $T$  of  $D$  with  $p$  leaves. As in the proof of Lemma 1, we obtain a collection  $\mathcal{P}$  of  $p$  ( $< k$ ) vertex-disjoint directed paths covering all vertices of  $D$ .

For a path  $P \in \mathcal{P}$ , let  $W(P)$  be the set of vertices not on  $P$  which are out-neighbors of vertices on  $P$ . If  $|W(P)| \geq k$ , then the vertices  $P$  and  $W(P)$  would form an out-tree with at least  $k$  leaves, which by the definition of  $\mathcal{L}$ , contradicts the assumption  $\ell_s(D) < k$ . Therefore,  $|W(P)| < k$ . We define

$$U_1 = \{v \in W(P) : P \in \mathcal{P}\}.$$

Note that

$$|U_1| \leq p(k - 1) \leq (k - 1)^2.$$

Let  $D_1$  be the graph obtained from  $D$  after applying the following trimming procedure around all vertices of  $U_1$ : for every path  $P \in \mathcal{P}$  and every vertex  $v \in U_1 \cap V(P)$  we delete all arcs emanating out of  $v$  and directed into  $v$  except those of the path  $P$  itself. Thus for every two paths  $P, Q \in \mathcal{P}$  there is no arc in  $D_1$  that goes from  $P$  to  $Q$ .

For  $P \in \mathcal{P}$  let  $D_1[P]$  be the subgraph of  $D_1$  induced by the vertices of  $P$ . Observe that  $P$  is a Hamiltonian directed path in  $D_1[P]$  and the connectivity components of  $D_1$  are the induced subgraphs of  $D_1$  on the paths  $P$  for  $P \in \mathcal{P}$ .

Let  $P \in \mathcal{P}$ , we will show that the  $pw(UN(D_1[P]))$  is bounded by  $k^2 - 2k + 2$ . We denote by  $S[P]$  the set of vertices which are heads of forward arcs in  $D_1[P]$ .

We claim that  $|S[P]| \leq (k - 2)(k - 1)$ . Indeed, for each vertex  $v \in S[P]$ , delete all forward arcs terminating at  $v$  but one. Observe that the procedure has not changed the number of vertices which are heads of forward arcs. Also the number of forward arcs in the new digraph is  $|S[P]|$ . As in the proof of Lemma 11, we can show that the number of forward arcs in the new digraph is at most  $(k - 2)(k - 1)$ .

Let  $D_2[P]$  be the graph obtained from  $D_1[P]$  after applying the trimming procedure as before around all vertices of  $S[P]$ , that is, for every vertex  $v \in S[P]$  we delete all arcs emanating out of  $v$  or directed into  $v$  except those of the path  $P$ .

Observe that  $D_2[P]$  consists of the directed path  $P = v_1v_2 \dots v_q$  passing through all its vertices, together with its backward arcs. For every  $j \in [q]$  let  $V_j = \{v_i : i \in [j]\}$ . If for some  $j$  the set  $V_j$  contained  $k$  vertices, say  $\{v'_1, v'_2, \dots, v'_k\}$ , having in-neighbors in the set  $\{v_{j+1}, v_{j+2}, \dots, v_q\}$ , then  $D$  would contain an out-tree with  $k$  leaves formed by the path  $v_{j+1}v_{j+2} \dots v_q$  together with a backward arc terminating at  $v'_i$  from a vertex on the path for each  $1 \leq i \leq k$ , a contradiction. Thus  $vs(UN(D_2[P])) \leq k$ . By Proposition 2, the pathwidth of  $UN(D_2[P])$  is at most  $k$ . Let  $(X_1, X_2, \dots, X_p)$  be a path decomposition of  $UN(D_2[P])$  of width at most  $k$ . Then  $(X_1 \cup S[P], X_2 \cup S[P], \dots, X_p \cup S[P])$  is a path decomposition of  $UN(D_1[P])$  of width at most  $k + |S[P]| \leq k^2 - 2k + 2$ .

The pathwidth of a graph is equal to the maximum pathwidth of its connected components. Hence, there exists a path decomposition  $(X_1, X_2, \dots, X_q)$  of  $UN(D_1)$  of width at most  $k^2 - 2k + 2$ . Then  $(X_1 \cup U_1, X_2 \cup U_1, \dots, X_q \cup U_1)$  is a path decomposition of  $UN(D)$ . Thus, the pathwidth of the underlying graph of  $D$  is at most  $k^2 - 2k + 2 + |U_1| \leq k^2 - 2k + 2 + (k - 1)^2 \leq 2k^2$ .  $\square$

**Theorem 3.**  *$k$ -DMLOB is FPT for digraphs in  $\mathcal{L}$ .*

*Proof.* Let  $D$  be a digraph in  $\mathcal{L}$  with  $\ell_s(D) > 0$  and  $n$  vertices. The proof of Theorem 2 can be easily turned into a polynomial time algorithm to either build an out-branching of  $D$  with at least  $k$  leaves or to show that  $pw(UN(D)) \leq 2k^2$  and provide the corresponding path decomposition. A simple dynamic programming over the decomposition gives us an algorithm of running time  $O(k^{O(k^2)} \cdot n^{O(1)})$ . Alternatively, the property of containing a directed out-branching with at least  $k$  leaves can be formulated as a monadic second order formula. Thus, by the fundamental theorem of Courcelle [8,9], the  $k$ -DMLOB problem for all digraphs  $D$



with  $pw(UN(D)) \leq 2k^2$  can be solved in  $O(f(k) \cdot n)$  time, where  $f$  is a function depending only on  $k$ .  $\square$

Let  $D$  be a digraph and let  $R_v$  be the set of vertices reachable from a vertex  $v \in V(D)$  in  $D$ . Observe that  $D$  has an out-tree with  $k$  leaves if and only if there exists a  $v \in V(D)$  such that  $D[R_v]$  has an out-tree with  $k$  leaves. Notice that each  $D[R_v]$  has an out-branching rooted at  $v$ . Thus, we can prove the following theorem, using the arguments in the previous proofs.

**Theorem 4.** *For a digraph  $D$  and  $v \in V(D)$ , let  $R_v$  be the set of vertices reachable from a vertex  $v \in V(D)$  in  $D$ . Then either we have  $\ell(D[R_v]) \geq k$  or the underlying undirected graph of  $D[R_v]$  is of pathwidth at most  $2k^2$ . Moreover, one can find, in polynomial time, either an out-tree with at least  $k$  leaves in  $D[R_v]$ , or a path decomposition of it of width at most  $2k^2$ .*

To solve  $k$ -DMLOT, we apply Theorem 4 to all the vertices of  $D$  and then either apply dynamic programming over the decomposition or apply Courcelle's Theorem as in the proof of Theorem 3. This gives the following:

**Theorem 5.**  *$k$ -DMLOT is FPT for digraphs.*

We can, in fact, show that the  $k$ -DMLOB problem for digraphs in  $\mathcal{L}$  is linear time solvable for a fixed  $k$ . To do so, given a digraph  $D \in \mathcal{L}$  with  $\ell_s(D) > 0$  we first apply Bodlaender's linear time algorithm [4] to check whether the treewidth of  $UN(D)$  is at most  $2k^2$ . If  $tw(UN(D)) > 2k^2$  then by Theorem 2  $D$  has an out-branching with at least  $k$  leaves. Else  $tw(UN(D)) \leq 2k^2$  and we can use Courcelle's Theorem to check in linear time whether  $D$  has an out-branching with at least  $k$  leaves. This gives the following:

**Theorem 6.** *The  $k$ -DMLOB problem for digraphs in  $\mathcal{L}$  is linear time solvable for every fixed  $k$ .*

## 5 Concluding Remarks and Open Problems

We have shown that every digraph  $D \in \mathcal{L}$  with  $\ell_s(D) > 0$  of order  $n$  and with minimum in-degree at least 3 contains an out-branching with at least  $(n/2)^{1/5} - 1$  leaves. Combining the ideas in the proof of this combinatorial result with the fact that the problem of deciding whether a given digraph in  $\mathcal{L}$  has an out-branching with at least  $k$  leaves can be solved efficiently for digraphs of pathwidth at most  $2k^2$  we have shown that the  $k$ -DMLOB problem for digraphs in  $\mathcal{L}$  as well as the  $k$ -DMLOT problem for general digraphs are fixed parameter tractable. The parameterized complexity of the  $k$ -DMLOB problem for all digraphs remains open.

For some subfamilies of  $\mathcal{L}$ , one can obtain better bounds on  $\ell_s(D)$ . An example is the class of multipartite tournaments. A multipartite tournament is an orientation of a complete multipartite graph. It is proved in [20,29] that every multipartite tournament  $D$  with at most one source has an out-branching  $T$  such

that the distance from the root of  $T$  to any vertex is at most 4. This implies that  $\ell_s(D) \geq \frac{n-1}{4}$ . Also for a tournament  $D$  of order  $n$ , it is easy to prove that  $\ell_s(D) \geq n - \log_2 n$ . (This bound is essentially tight, i.e., we cannot replace the right hand side by  $n - \log_2 n + \Omega(\log_2 \log_2 n)$  as shown by random tournaments; see [1], pages 3-4, for more details.)

It seems that the bound  $\ell_s(D) \geq (n/2)^{1/5} - 1$  is far from tight. It would be interesting to obtain better bounds for digraphs  $D \in \mathcal{L}$  (with  $\ell_s(D) > 0$ ) of minimum in-degree at least 3.

**Acknowledgements.** We thank Bruno Courcelle, Martin Grohe, Eun Jung Kim and Stephan Kreutzer for useful discussions of the paper. Research of Noga Alon and Michael Krivelevich was supported in part by a USA-Israeli BSF grant and by a grant from the Israel Science Foundation. Research of Fedor Fomin was supported in part by the Norwegian Research Council. Research of Gregory Gutin was supported in part by an EPSRC grant.

## References

1. Alon, N., Spencer, J.: The Probabilistic Method, 2nd edn. Wiley, Chichester (2000)
2. Bang-Jensen, J., Gutin, G.: Digraphs: Theory, Algorithms and Applications. Springer, Heidelberg (2000)
3. Bodlaender, H.L.: On linear time minor tests and depth-first search. *Journal of Algorithms* 14, 1–23 (1993)
4. Bodlaender, H.L.: A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM Journal on Computing* 25, 1305–1317 (1996)
5. Bonsma, P.S., Brueggermann, T., Woeginger, G.J.: A faster FPT algorithm for finding spanning trees with many leaves. In: Rován, B., Vojtáš, P. (eds.) MFCS 2003. LNCS, vol. 2747, pp. 259–268. Springer, Heidelberg (2003)
6. Cesati, M.: Compendium of parameterized problems (September 2006), <http://bravo.ce.uniroma2.it/home/cesati/research/compendium.pdf>
7. Estivill-Castro, V., Fellows, M.R., Langston, M.A., Rosamond, F.A.: FPT is P-Time Extremal Structure I. In: Proc. ACiD, pp. 1–41 (2005)
8. Courcelle, B.: The Monadic second-order logic of graphs I: recognizable sets of finite graphs. *Information and Computation* 85, 12–75 (1990)
9. Courcelle, B.: The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. *Informatique Théorique et Applications (ITA)* 26, 257–286 (1992)
10. Ding, G., Johnson, T., Seymour, P.: Spanning trees with many leaves. *Journal of Graph Theory* 37, 189–197 (2001)
11. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
12. Fellows, M.R., Langston, M.A.: On well-partial-order theory and its applications to combinatorial problems of VLSI design. *SIAM Journal on Discrete Mathematics* 5, 117–126 (1992)
13. Fellows, M.R., McCartin, C., Rosamond, F.A., Stege, U.: Coordinated kernels and catalytic reductions: An improved FPT algorithm for max leaf spanning tree and other problems. In: Kapoor, S., Prasad, S. (eds.) FST TCS 2000: Foundations of Software Technology and Theoretical Science. LNCS, vol. 1974, pp. 240–251. Springer, Heidelberg (2000)

14. Fellows, M.: Private communications (2005-2006)
15. Flum, J., Grohe, M.: *Parameterized Complexity Theory*. Springer, Heidelberg (2006)
16. Galbiati, G., Maffioli, F., Morzenti, A.: A short note on the approximability of the maximum leaves spanning tree problem. *Information Processing Letters* 52, 45–49 (1994)
17. Galbiati, G., Morzenti, A., Maffioli, F.: On the approximability of some maximum spanning tree problems. *Theoretical Computer Science* 181, 107–118 (1997)
18. Garey, M.R., Johnson, D.S.: *Computers and Intractability*. W.H. Freeman and Co, New York (1979)
19. Griggs, J.R., Wu, M.: Spanning trees in graphs of minimum degree four or five. *Discrete Mathematics* 104, 167–183 (1992)
20. Gutin, G.: The radii of  $n$ -partite tournaments. *Math. Notes* 40, 743–744 (1986)
21. Gutin, G., Yeo, A.: Some Parameterized Problems on Digraphs. To appear in *The Computer Journal*
22. Kleitman, D.J., West, D.B.: Spanning trees with many leaves. *SIAM Journal on Discrete Mathematics* 4, 99–106 (1991)
23. Kinnersley, N.G.: The vertex separation number of a graph equals its path-width. *Complementary Definitions of Programming Language Semantics* 42, 345–350 (1992)
24. Kirousis, L.M., Papadimitriou, C.H.: Interval graphs and searching. *Mes premieres constructions de programmes* 55, 181–184 (1985)
25. Linial, N., Sturtevant, D.: Unpublished result (1987)
26. Lu, H.-I., Ravi, R.: Approximating maximum leaf spanning trees in almost linear time. *Interval Mathematics* 29, 132–141 (1998)
27. Möhring, R.H.: Graph problems related to gate matrix layout and PLA folding. *Computational Graph Theory 7(Comput. Suppl.)*, 17–51 (1990)
28. Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, oxford (2006)
29. Petrovic, V., Thomassen, C.: Kings in  $k$ -partite tournaments. *Discrete Mathematics* 98, 237–238 (1991)
30. Solis-Oba, R.: 2-approximation algorithm for finding a spanning tree with the maximum number of leaves. In: Bilardi, G., Pietracaprina, A., Italiano, G.F., Pucci, G. (eds.) *ESA 1998*. LNCS, vol. 1461, pp. 441–452. Springer, Heidelberg (1998)
31. Robertson, N., Seymour, P.D.: Graph minors—a survey. In: Anderson, I. (ed.) *Surveys in Combinatorics*, pp. 153–171. Cambridge Univ. Press, Cambridge (1985)
32. Robertson, N., Seymour, P.D.: Graph minors I: Excluding a forest. *Journal of Combinatorial Theory Series B* 35, 39–61 (1983)
33. Robertson, N., Seymour, P.D.: Graph minors II: Algorithmic aspects of tree-width. *Journal of Algorithms* 7, 309–322 (1986)
34. Wu, B.Y., Chao, K.: *Spanning Trees and Optimization Problems*, CRC Press (2003)

# Parameterized Approximability of the Disjoint Cycle Problem

Martin Grohe and Magdalena Grüber

Institut für Informatik, Humboldt-Universität, Unter den Linden 6, 10099 Berlin,  
Germany

{grohe,grueber}@informatik.hu-berlin.de

**Abstract.** We give an fpt approximation algorithm for the directed vertex disjoint cycle problem. Given a directed graph  $G$  with  $n$  vertices and a positive integer  $k$ , the algorithm constructs a family of at least  $k/\rho(k)$  disjoint cycles of  $G$  if the graph  $G$  has a family of at least  $k$  disjoint cycles (and otherwise may still produce a solution, or just report failure). Here  $\rho$  is a computable function such that  $k/\rho(k)$  is nondecreasing and unbounded. The running time of our algorithm is polynomial.

The directed vertex disjoint cycle problem is hard for the parameterized complexity class  $W[1]$ , and to the best of our knowledge our algorithm is the first fpt approximation algorithm for a natural  $W[1]$ -hard problem.

**Keywords:** approximation algorithms, fixed-parameter tractability, parameterized complexity theory.

## 1 Introduction

Fixed-parameter tractability and approximability are two ways of dealing with intractability. Fixed-parameter tractability is a relaxed notion of tractability based on a “two-dimensional” complexity measure that not only depends on the size of the input instances, but also on an additional parameter, for example, the tree width of a graph, or the number of variables of a logical formula. Intuitively, a problem is fixed-parameter tractable if it can be solved efficiently on instances with a small value of the parameter. Formally, a parameterized problem is *fixed parameter tractable* if there is an algorithm solving the problem with a running time  $f(k) \cdot n^{O(1)}$ , where  $f$  is a computable function,  $k$  denotes the parameter value and  $n$  the size of the input. We call an algorithm achieving such a running time an *fpt algorithm*. Optimization problems are often parameterized by the cost of the solution. For example, the *disjoint cycle problem*<sup>1</sup> asks for a maximum family of vertex disjoint cycles in a given directed graph. In the parameterized version, the objective is to find a family of at least  $k$  disjoint cycles, where  $k$  is the parameter. Parameterized complexity theory gives evidence for the disjoint cycle problem not being fixed-parameter tractable; the problem is hard for the complexity class

---

<sup>1</sup> More precisely, we call the problem the *maximum directed vertex disjoint cycle problem*. It is also known as the (*maximum directed*) *cycle packing problem*.

W[1] (this follows easily from the results of [20]). Now we may ask if the problem is at least “approximately fixed parameter tractable”, that is, if there is an fpt algorithm that finds a family of disjoint cycles of size approximately  $k$  if the input graph has a family of  $k$  disjoint cycles. Depending on the desired approximation ratio, “approximately  $k$ ” may mean, for example,  $k/2$  or maybe just  $\log k$ .

The notion of parameterized approximability has been introduced in three independent papers that all appeared last year [3,4,6]; also see the recent survey [14]. An fpt algorithm for a parameterized maximization problem is an *fpt approximation algorithm* with *approximation ratio*  $\rho$  if given an instance of the problem and a positive integer  $k$  it produces a solution of cost at least  $k/\rho(k)$  if the instance has a solution of size at least  $k$ . (If the instance has no solution of size  $k$ , then the output may be arbitrary.) Here  $\rho$  is a computable function such that  $k/\rho(k)$  is nondecreasing and unbounded. An analogous definition can be given for minimization problems. We observe that whenever a maximization problem has an fpt approximation algorithm with approximation ratio  $\rho$ , then it also has a polynomial time approximation algorithm, albeit with a worse ratio  $\rho' \geq \rho$  (but still only depending on  $k$ ). [4,6] give several examples of optimization problems that are not fpt approximable (under standard assumptions from parameterized complexity theory). Moreover, it was shown in [4] that every intractable parameterized problem is equivalent, under suitable reductions, to an (artificial) problem that is fpt approximable and to a problem that is fpt inapproximable. However, there are few examples of natural problems that are known to be fpt approximable, but not known to be fixed-parameter tractable.

The main result of this paper is an fpt approximation algorithm for the disjoint cycle problem. To the best of our knowledge, this is the first fpt approximation algorithm for a natural W[1]-hard problem. Our result is mainly of theoretical interest, because the approximation ratio is very close to  $k$ : We can only lower bound the number  $k/\rho(k)$  of disjoint cycles our algorithm is guaranteed to compute by a multiply iterated logarithm.

The disjoint cycle problem is well known to be NP-complete (see [2]) and recently, Salavatipour and Verstraete [18] proved that this problem is hard to approximate within a factor  $\Omega(\log^{1-\epsilon} n)$  unless  $\text{NP} \subseteq \text{DTIME}(2^{\text{poly} \log(n)})$ . They also proved that the problem has a polynomial time approximation algorithm with ratio  $O(\sqrt{n})$ . Note that this approximation algorithm is incomparable with ours, as the ratio is expressed in terms of  $n$ , which may be much larger than  $k$ .

The linear programming dual of the disjoint cycle problem is the directed feedback vertex set problem. It asks for a minimum set of vertices of a directed graph such that the graph obtained by deleting these vertices is acyclic. It is one of the most prominent open problems in parameterized complexity theory whether the directed feedback vertex set problem is fixed-parameter tractable. Based on a result by Seymour [19], Even et al. [8] gave a polynomial time algorithm that approximates the minimum feedback vertex set to a factor of  $O(\log k \cdot \log \log k)$ ; hence in particular the problem is fpt approximable. The size  $\tau(G)$  of the minimum feedback vertex set and the size  $\nu(G)$  of a maximum family of disjoint cycles are two graph invariants that have also received considerable attention in

graph theory (e.g. [12,7,12,13,17]). Clearly,  $\nu(G) \leq \tau(G)$ , and inequality may hold (for example,  $\nu(K_4) = 1 < 2 = \tau(K_4)$ ). In 1996, Reed, Robertson, Seymour and Thomas [16] proved that  $\tau(G)$  can be bounded in terms of  $\nu(G)$ ; this settled a long standing open conjecture due to Younger [21]. As  $\tau(G)$  can be approximated up to a logarithmic factor in polynomial time, it follows that there is a polynomial time algorithm that approximates  $\nu(G)$  with some approximation ratio  $\rho$  (not depending on the size of the graph; see [4] for details). However, this algorithm just computes an approximation of the solution size and not an actual solution, that is, a family of pairwise disjoint cycles of that size. This is what we achieve here. Let us remark that in the last section of [16], Reed et al. studied the algorithmic question of computing a maximum family of disjoint cycles and gave an algorithm that computes a family of  $k$  disjoint cycles in time  $n^{f(k)}$ , for some function  $f$ . But this algorithm is not an fpt algorithm.

Our algorithm is heavily based on [16]; indeed in large parts it may be viewed as an algorithmic version of the construction given there. However, there is a crucial step in the construction of [16] that cannot be made algorithmic so easily, or more precisely, cannot be turned into an fpt algorithm: Reed et al. start their construction by taking a minimum feedback vertex set. We do not know how to compute such a set by an fpt algorithm. Instead, we take an approximately minimum feedback vertex set, but then a rather straightforward argument by Reed et al. has to be turned into a complicated recursive algorithm. We will explain this algorithm in detail in Section 5. Due to space limitations, we have to omit most other details and proofs.

## 2 Preliminaries

$\mathbb{N}$  denotes the set of positive integers and  $\mathbb{R}$  the set of reals. A function  $f : \mathbb{N} \rightarrow \mathbb{R}$  is *nondecreasing* (*increasing*) if for all  $m, n \in \mathbb{N}$  with  $m < n$  it holds that  $f(m) \leq f(n)$  ( $f(m) < f(n)$ , respectively).  $f$  is *unbounded* if for all  $k \in \mathbb{N}$  there exists an  $n \in \mathbb{N}$  such that  $f(n) \geq k$ .

For a graph  $G$  the set of vertices (edges) is denoted by  $V(G)$  ( $E(G)$ ) and the number of vertices is denoted by  $|G|$ . For a subset  $X \subseteq V(G)$ , by  $G[X]$  we denote the induced subgraph of  $G$  with vertex set  $X$ , and by  $G \setminus X$  the subgraph  $G[V(G) \setminus X]$ . Graphs are always directed. *Paths* are directed and have no repeated vertices, and an  $(a, b)$ -*path* in a graph  $G$  is a path from  $a$  to  $b$ . A *linkage*  $L$  in a directed graph  $G$  is a set of vertex disjoint paths. A linkage  $L$  consisting of paths  $P_1, \dots, P_k$ , where  $P_i$  is an  $(a_i, b_i)$ -path ( $1 \leq i \leq k$ ), is said to *link*  $(a_1, \dots, a_k)$  to  $(b_1, \dots, b_k)$ . For  $A, B \subseteq V(G)$  with  $a_1, \dots, a_k \in A$  and  $b_1, \dots, b_k \in B$  we say that  $L$  is a linkage *from*  $A$  *to*  $B$  *of size*  $k$ . A pair  $(X, Y)$  is a *separation* for a directed graph  $G$  (of *order*  $|X \cap Y|$ ) if  $X, Y \subseteq V(G)$  with  $X \cup Y = V(G)$  and if no edge of  $G$  has tail in  $X \setminus Y$  and head in  $Y \setminus X$ . Menger's theorem states that for all directed graphs  $G$ , sets  $A, B \subseteq V(G)$ , and  $k \geq 1$  there is a linkage of size  $k$  from  $A$  to  $B$  if and only if there is no separation  $(X, Y)$  of  $G$  of order  $< k$  with  $A \subseteq X$  and  $B \subseteq Y$ . We will use the following algorithmic version, which can be proved by standard network flow techniques:

**Fact 1.** *There exists a polynomial time algorithm that given a directed graph  $G$ , a positive integer  $k$  and two subsets  $A, B \subseteq V(G)$  both of size  $\geq k$  computes either  $k$  many vertex disjoint paths from  $A$  to  $B$  or a separation  $(X, Y)$  of order  $< k$  with  $A \subseteq X, B \subseteq Y$ .*

The *Ramsey number*  $R_l(m, q)$  is the minimum integer such that for any set of size  $\geq R_l(m, q)$  if all subsets of size  $l$  are colored with  $q$  colors, then there exists a subset of size  $m$  such that all its size- $l$  subsets have the same color. A simple upper bound is

$$R_l(m, q) \leq \exp^{(l)}(c_q \cdot m) \tag{1}$$

for some positive constant  $c_q$ , where the iterated exponential function  $\exp^{(l)}$  is defined inductively as  $\exp^{(1)}(x) = x$  and  $\exp^{(l)}(x) := 2^{\exp^{(l-1)}(x)}$  for  $l \geq 2$  [10].

### 3 Disjoint Cycles and Feedback Vertex Sets

Recall that the *directed maximum vertex disjoint cycle problem* (for short: *disjoint cycle problem*) is the maximization problem whose objective it is to find a family of pairwise vertex disjoint cycles of maximum size in a given directed graph. For a directed graph  $G$ , we denote the maximum size of a family of pairwise vertex disjoint cycles by  $\nu(G)$ .

The linear programming dual of the disjoint cycle problem is the *directed minimum feedback vertex set problem*. A *feedback vertex set* in a directed graph  $G$  is a set  $S$  of vertices such that the graph  $G \setminus S$  is acyclic. The objective of the directed minimum feedback vertex set problem is to find a feedback vertex set of minimum size in a given graph. We denote the minimum size of a feedback vertex set of a directed graph  $G$  by  $\tau(G)$ . Obviously, we have  $\nu(G) \leq \tau(G)$ . For a given graph  $G = (V, E)$ , consider the following linear program with variables  $x_v$  for the vertices  $v \in V$ :

$$\begin{aligned} \text{Minimize } \sum_{v \in V} x_v \quad & \text{subject to } \sum_{v \in C} x_v \geq 1 \text{ for every cycle } C \text{ in } \mathcal{G}, \\ & x_v \geq 0 \quad \text{for every vertex } v \in V. \end{aligned} \tag{2}$$

The  $\{0, 1\}$ -solutions correspond to the feedback vertex sets of  $G$ . The rational solutions are called *fractional feedback vertex sets*, and the minimum cost  $\sum_{v \in V} x_v$  of a fractional feedback vertex set is denoted by  $\tau^*(G)$ . By standard techniques [11], a minimum fractional feedback vertex set and the parameter  $\tau^*(G)$  can be computed in polynomial time (also see [8]). The natural linear programming formulation of the disjoint cycle problem yields the dual of the linear program (2). Hence the cost  $\nu^*(G)$  of an optimal fractional solutions coincides with  $\tau^*(G)$ , and we have

$$\nu(G) \leq \nu^*(G) = \tau^*(G) \leq \tau(G).$$

Let us state the main known results regarding the parameters  $\tau$  and  $\nu$ :

**Theorem 2 (Seymour [19], Even et al. [8]).** *For every directed graph  $G$  it holds that  $\tau(G) = O(\tau^*(G) \cdot \log(\tau^*(G)) \cdot \log \log(\tau^*(G)))$ . Furthermore, there is*

a polynomial time algorithm that constructs a feedback vertex set for a directed graph  $G$  of size at most  $O(\tau^*(G) \cdot \log(\tau^*(G)) \cdot \log \log(\tau^*(G)))$ .

**Theorem 3 (Reed et al. [16]).** *There exists a computable function  $f$  such that for all directed graphs  $G$  it holds that  $\tau(G) \leq f(\nu(G))$ .*

The function  $f$  constructed by Reed et al. [16] grows very quickly. It is a multiply iterated exponential, where the number of iterations is also a multiply iterated exponential.

## 4 Fixed Parameter Tractable Approximation Algorithms

For background in parameterized complexity theory, we refer the reader to [5,9,15]. We only define the notions needed here. Recall the definition of fpt algorithms from the introduction. Intuitively, an fpt approximation algorithm is an algorithm whose running time is fpt for the parameter “cost of the solution” and whose approximation ratio only depends on the parameter and not on the size of the input. Hence every polynomial time approximation algorithm with constant approximation ratio is an fpt approximation algorithm, but an approximation algorithm with approximation ratio  $\log n$ , where  $n$  denotes the input size, is not.

For simplicity, we only define fpt approximation algorithms for maximization problems. The definition can easily be adapted for minimization problems.

**Definition 4.** Let  $\rho : \mathbb{N} \rightarrow \mathbb{N}$  be a computable function such that  $k/\rho(k)$  is nondecreasing and unbounded. An *fpt approximation algorithm* for an NP-maximization problem  $O$  (over some alphabet  $\Sigma$ ) with *approximation ratio*  $\rho$  is an algorithm  $\mathbb{A}$  with the following properties:

1.  $\mathbb{A}$  expects inputs  $(x, k) \in \Sigma^* \times \mathbb{N}$ . For every input  $(x, k) \in \Sigma^* \times \mathbb{N}$  such that there exists a solution for  $x$  of cost at least  $k$ , the algorithm  $\mathbb{A}$  computes a solution for  $x$  of cost at least  $k/\rho(k)$ . For inputs  $(x, k) \in \Sigma^* \times \mathbb{N}$  without solution of cost at least  $k$ , the output of  $\mathbb{A}$  can be arbitrary.
2. There exists a computable function  $f$  such that the running time of  $\mathbb{A}$  on input  $(x, k)$  is bounded by  $f(k) \cdot |x|^{O(1)}$ .

$O$  is *fpt approximable* if there exists an fpt approximation algorithm for  $O$  with ratio  $\rho$ , for some computable function  $\rho : \mathbb{N} \rightarrow \mathbb{N}$  such that  $k/\rho(k)$  is nondecreasing and unbounded.

The following lemma, which may be surprising at first sight, but is actually quite simple, states that we can make the running time of an fpt approximation algorithm polynomial at the cost of a worse approximation ratio. Note that this is not something one would usually do in practice, but it is relevant theoretically because it shows that the notion of fpt approximability is fairly robust.

**Lemma 5.** *For every fpt approximable maximization problem  $O$  there exists a polynomial time algorithm  $\mathbb{A}$  that is an fpt approximation algorithm for  $O$ .*



In other words, we can replace requirement (2) in Definition 4 by the stronger requirement (2') below and obtain an equivalent notion of fpt approximability:

(2') The running time of  $\mathbb{A}$  on input  $(x, k)$  is  $(|x| + k)^{O(1)}$ .

Let us remark that the analogue of Lemma 5 for minimization problems does not hold.

We also need the following definition to establish a relationship between approximation algorithms that are efficient for small optima and fpt approximation algorithms. For an instance  $x$  of an NP-optimization problem  $O$ , by  $\text{opt}(x)$  we denote the cost of an optimal solution.

**Definition 6.** An NP-maximization problem  $O$  is *well-behaved for parameterized approximation* if there exists a constant  $c \geq 0$  such that the following holds:

1. Given an instance  $x \neq \epsilon$  for  $O$  it is possible to construct a new instance  $x'$  for  $O$  in time polynomial in  $|x|$  such that  $|x'| < |x|$  and  $\text{opt}(x) \geq \text{opt}(x') \geq \text{opt}(x) - c$ . (Here we assume that the empty string  $\epsilon$  is an instance for  $O$ .)
2. For every instance  $x \neq \epsilon$  it holds that a valid solution for the constructed instance  $x'$  is also a valid solution for  $x$ .

For example, the disjoint cycle problem is well-behaved, because given a graph  $G$  we can delete a vertex  $w$  and obtain a graph  $G' = G \setminus \{w\}$  such that  $\nu(G) \geq \nu(G') \geq \nu(G) - 1$ , and every family of disjoint cycles of  $G'$  is also a family of disjoint cycles of  $G$ .

**Proposition 7 (Chen et al. 4).** *Let  $O$  be an NP-maximization problem over the alphabet  $\Sigma$  that is well-behaved for parameterized approximation, and let  $\rho : \mathbb{N} \rightarrow \mathbb{N}$  be a computable function such that  $k/\rho(k)$  is nondecreasing and unbounded. If there exists a computable function  $g$  and an algorithm  $\mathbb{B}$  that given an input  $x \in \Sigma^*$  computes a solution  $y$  for  $x$  of cost  $k$  such that  $k \geq \text{opt}(x)/\rho(\text{opt}(x))$  in time  $g(\text{opt}(x)) \cdot |x|^{O(1)}$ , then  $O$  has an fpt approximation algorithm with approximation ratio  $\rho$ .*

## 5 The Main Theorem

**Theorem 8.** *The directed maximum vertex disjoint cycle problem has an fpt approximation algorithm with polynomial running time.*

First, we will design an algorithm  $\mathbb{A}$  that computes at least  $\tau^*(G)/\rho(\tau^*(G))$  vertex disjoint cycles for a given directed graph  $G$  and some computable function  $\rho : \mathbb{N} \rightarrow \mathbb{N}$  with  $\frac{x}{\rho(x)}$  being nondecreasing and unbounded and such that the running time of this algorithm is bounded by  $g(\tau^*(G)) \cdot |G|^{O(1)}$  for some computable function  $g$ . We will then see why this also gives an fpt approximation algorithm for the disjoint cycle problem.

The core of our algorithm  $\mathbb{A}$  is a recursive procedure, described in Section 5.2 that either computes sufficiently many vertex disjoint cycles in a given graph

$G$  directly, or guarantees the existence of linkages between all subsets  $A, B$  of a certain size  $m$  of some feedback vertex set  $T$  of  $G$ . It remains to give an algorithm that computes sufficiently many vertex disjoint cycles if such linkages exist. This is what the “Linkage Lemma” (Lemma 10) achieves. Very roughly, the existence of the linkages guarantees the existence of a substructure of the graph called a “fence”, together with a linkage that connects the “bottom” with the “top” of the fence. In such a fence it is then possible to find the desired cycles.

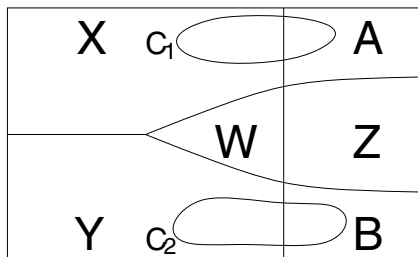
### 5.1 Technical Lemmas

**Lemma 9 (Splitting Lemma).** *Let  $G$  be a directed graph,  $T$  a feedback vertex set for  $G$ , and  $m \leq |T|/2$ . Then at least one of the following possibilities holds:*

- (i) *For all  $A, B \subseteq T$  with  $|A| = |B| = m$  there exists a linkage from  $A$  to  $B$  with no vertex in  $T \setminus (A \cup B)$ .*
- (ii) *There is a feedback vertex set  $T'$  of  $G$  with  $|T'| < |T|$ .*
- (iii) *There are two vertex disjoint subgraphs  $G_1$  and  $G_2$  of  $G$ , cycles  $C_1$  in  $G_1$  and  $C_2$  in  $G_2$ , and feedback vertex sets  $T_1$  for  $G_1$  and  $T_2$  for  $G_2$  such that:*
  - $|T_1| = |T_2| = m$ .
  - *For all feedback vertex sets  $T'_1$  of  $G_1$  and  $T'_2$  of  $G_2$  the set  $T'_1 \cup T'_2 \cup (V(G) \setminus V(G_1 \cup G_2))$  is a feedback vertex set of  $G$  of size at most  $|T'_1| + |T'_2| + |T| - (m + 1)$ .*

Furthermore, there is an algorithm that, given  $G, T$ , and  $m$ , either recognizes that (i) holds or computes a feedback vertex set  $T'$  as in (ii) or computes  $G_1, C_1, T_1, G_2, C_2, T_2$  as in (iii). The running time of the algorithm is  $3^{|T|} \cdot |G|^{O(1)}$ .

*Proof.* Suppose that (i) does not hold. Let  $A, B \subseteq T$  with  $|A| = |B| = m$  such that there is no linkage from  $A$  to  $B$  in  $G$  that has no vertex in  $Z := T \setminus (A \cup B)$ . By Menger’s theorem, there exists a separation  $(X, Y)$  of  $G$  with  $A \subseteq X, B \subseteq Y, Z \subseteq (X \cap Y)$  and with  $|W| < m$ , where  $W := X \cap Y \setminus Z$ . Let  $G_1$  be  $G \setminus Y$  and



**Fig. 1.** A sketch of the separation  $(X, Y)$  for  $G$  (assuming  $X \cap B = \emptyset$  and  $Y \cap A = \emptyset$ )

$G_2$  be  $G \setminus X$ . Furthermore let  $D_1 := A \cup (X \cap Y)$  and  $D_2 := B \cup (X \cap Y)$ .

Then it holds that  $|D_1| = |D_2| < |T|$ , because  $X \cap Y = W \cup Z, |Z| = |T \setminus (A \cup B)| = |T| - 2m$ , and  $|W| < m$ . It might happen that  $D_1$  or  $D_2$  is again

a feedback vertex set for  $G$ , in which case we succeeded at finding a smaller feedback vertex set for  $G$ . Thus (ii) holds. Otherwise, there are cycles  $C_1, C_2$  such that  $V(C_1) \cap D_2 = \emptyset$  and  $V(C_2) \cap D_1 = \emptyset$ . We claim that  $C_i \subseteq G_i$  for  $i = 1, 2$ . Let us look at  $C_1$ ; the proof for  $C_2$  is completely analogous. Since  $T$  is a feedback vertex set, we have  $T \cap V(C_1) \neq \emptyset$  and thus  $A \cap V(C_1) \neq \emptyset$ , because  $D_2 \cap V(C_1) = \emptyset$ . Since  $(X, Y)$  is a separation and  $X \cap Y \cap V(C_1) = \emptyset$ , it follows that  $V(C_1)$  does not meet  $Y$  and therefore the cycle  $C_1$  is contained in  $G_1$ .

Observe that  $A$  and  $B$  are feedback vertex sets for  $G_1$  and  $G_2$ , respectively, because every cycle must meet  $T$ , and  $T \cap V(G_1) = A$ . Hence we can let  $T_1 := A$  and  $T_2 := B$ . It is easy to check that (iii) is satisfied. It remains to prove the algorithmic statement. Using Fact [□](#), we can compute sets  $A, B$  and a separation  $(X, Y)$  with the properties above if such sets  $A, B$  exist in time  $3^{|T|} \cdot |G|^{O(1)}$ . Here we use  $3^{|T|}$  as an upper bound for the number of partitions of  $T$  into three sets  $A, B, Z$ . Given  $A, B, X, Y$ , we can check in polynomial time if the sets  $D_1, D_2$  defined above are feedback vertex sets and find the cycles  $C_1, C_2$  if not.  $\square$

We define  $\kappa = \kappa(G)$  for a given graph  $G$  to be the maximum integer with

$$(\kappa - 1)^2 + 1 + \exp^{((\kappa-1)^2+1)} (c_{((\kappa-1)^2+1)!+1})^2 \cdot (\kappa+1) \cdot ((\kappa - 1)^2 + 1) \leq \tau^*(G) \tag{3}$$

and let

$$\lambda(G) := (\kappa(G) - 1)^2 + 1 \quad \text{and} \quad \mu(G) := (\kappa(G) + 1) \cdot \lambda(G) . \tag{4}$$

**Lemma 10 (Linkage Lemma).** *There is a computable, nondecreasing and unbounded function  $\varphi$  such that the following holds: Let  $G$  be a directed graph,  $m \leq \mu(G)$ , and let  $T$  be a feedback vertex set of  $G$  such that for all  $A, B \subseteq T$  with  $|A| = |B| = m$  there exists a linkage from  $A$  to  $B$  with no vertex in  $T \setminus (A \cup B)$ . Then there exists a family of at least  $\varphi(m)$  vertex disjoint cycles in  $G$ . Furthermore, there is an algorithm that computes such a family in time  $g(|T|) \cdot |G|^{O(1)}$ , for some computable function  $g$ .*

The proof of this lemma is based on algorithmic versions of Theorems 2.2., 2.3. and 2.4. of [\[16\]](#). It will appear in the full version of this paper.

### 5.2 The Main Algorithm

**Lemma 11.** *There is a computable, nondecreasing, unbounded function  $\psi$ , a computable function  $g$ , and an algorithm  $\mathbb{A}$  such that the following holds: Given a directed graph  $G$ , the algorithm  $\mathbb{A}$  computes a family of at least  $\psi(\tau^*(G))$  vertex disjoint cycles of  $G$  in time  $g(\tau^*(G)) \cdot |G|^{O(1)}$*

*Proof.* Let  $G$  be a directed graph. Set  $m_i := \lfloor \mu(G)/4^i \rfloor$  for  $i \geq 0$  and let  $i^*$  be the minimal integer such that  $m_i \leq 4$ . Let  $\varphi$  be the function from the Linkage Lemma (Lemma [□0](#)). We define a function  $\chi$  by

$$\chi(0) := \chi(1) := \min\{\varphi(1), i^*\}, \quad \chi(k) := \min \left\{ \chi \left( \left\lfloor \frac{k}{4} \right\rfloor \right) + 1, \varphi(k), i^* \right\} .$$

It is easy to see that  $\chi$  is computable, nondecreasing, and unbounded. By definition of  $\mu(G)$  in (4), there is a computable, nondecreasing and unbounded function  $\zeta$  such that  $\zeta(\tau^*(G)) = \mu(G)$  for all directed graphs  $G$ . We define the function  $\psi$  by  $\psi(x) := \chi(\zeta(x))$ .

On input  $G$ , algorithm  $\mathbb{A}$  first computes a feedback vertex set  $T$  of  $G$  of size  $O(\tau^*(G) \cdot \log \tau^*(G) \cdot \log \log \tau^*(G))$  using the algorithm of Theorem 2. Then  $\mathbb{A}$  builds up a binary tree  $\mathcal{B}$ . The vertices of this tree are labelled by pairs  $(G', T')$ , where  $G'$  is a graph and  $T'$  a feedback vertex set of  $G'$ . The edges of the tree  $\mathcal{B}$  are labelled by cycles of  $G$ .

At each point during the execution, the algorithm  $\mathbb{A}$  processes a leaf of the tree and then either halts or modifies the tree and continues with some other node. If we think of the algorithm  $\mathbb{A}$  as a recursive algorithm, then  $\mathcal{B}$  is just a convenient way to describe the content of the stack during the execution of  $\mathbb{A}$ .

Initially, the tree  $\mathcal{B}$  only consists of its root, which is labelled by the pair  $(G, T)$ . The algorithm starts at this root. Now suppose at some stage during its execution the algorithm has to process a leaf  $b$  labelled by  $(G_b, T_b)$  and of height  $i$ . (The *height* of  $b$  is defined to be the length of the path from the root to  $b$ .) If  $i \geq i^*$ , the algorithm  $\mathbb{A}$  halts and outputs the cycles labelling the edges of the path from the root to  $b$ . Otherwise, it calls the algorithm of the Splitting Lemma (Lemma 9) with input  $G_b, T_b$ , and  $m_i$ . (We will prove later that the assumption  $2m_i \leq T_b$  is satisfied.) According to the three outcomes of the Splitting Lemma, we have to distinguish the following cases:

**Linkage Step.** If for all  $A, B \subseteq T_b$  with  $|A| = |B| = m_i$  there exists a linkage in  $G_b$  from  $A$  to  $B$  with no vertex in  $T_b \setminus (A \cup B)$ , then the algorithm of the Linkage Lemma is called with input  $(G_b, T_b)$ . It returns a family  $\mathcal{C}$  of  $\varphi(m_i)$  pairwise disjoint cycles of  $G_b$ . Algorithm  $\mathbb{A}$  halts and outputs the cycles in  $\mathcal{C}$  together with the cycles labelling the edges of the path from the root to  $b$  in  $\mathcal{B}$ .

**Splitting Step.** If the algorithm of the Splitting Lemma returns vertex disjoint subgraphs  $G_1, G_2$ , cycles  $C_1 \subseteq G_1$  and  $C_2 \subseteq G_2$ , and feedback vertex sets  $T_1$  of  $G_1$  and  $T_2$  of  $G_2$ , then our algorithm  $\mathbb{A}$  proceeds as follows: It creates two new children  $b_1, b_2$  of  $b$ . For  $i = 1, 2$ , child  $b_i$  is labelled  $(G_i, T_i)$ , and the edge from  $b$  to  $b_i$  is labelled by the cycle  $C_{2-i}$  (so that the cycle labelling the edge to  $b_i$  is disjoint from the graph labelling the node  $b_i$ ). Now the processing of  $b$  is completed, and algorithm  $\mathbb{A}$  continues with child  $b_1$ .

**Shrinking Step.** If the algorithm of the Splitting Lemma returns a feedback vertex set  $T'_b$  of  $G_b$  with  $|T'_b| < |T_b|$ , then our algorithm  $\mathbb{A}$  relabels the current leaf  $b$  by  $(G_b, T'_b)$ . Then it calls the following subroutine  $\mathbb{S}$  at leaf  $b$ .

**Subroutine  $\mathbb{S}$ .** The subroutine  $\mathbb{S}$  proceeds as follows. Suppose it is at a leaf  $b_1$  of  $\mathcal{B}$  labelled  $(G_1, T_1)$ . If  $b_1$  is the root, the subroutine returns  $b_1$ . Otherwise, let  $b'$  be the parent and  $b_2$  the sibling of  $b_1$ . Suppose that  $b'$  is labelled  $(G', T')$  and  $b_2$  is labelled  $(G_2, T_2)$ . Let  $T'' = T_1 \cup T_2 \cup (V(G') \setminus (V(G_1 \cup G_2)))$ .  $\mathbb{S}$  distinguishes between three cases:

- If  $|T''| < |T'|$ , then  $\mathbb{S}$  deletes  $b_1$  and  $b_2$ . Then it relabels  $b'$  by  $(G', T'')$  and continues at node  $b'$ .

- If  $|T''| \geq |T'|$  and  $|T_2| > |T_1|$ , then  $\mathbb{S}$  returns  $b_2$ .
- Otherwise,  $\mathbb{S}$  returns  $b_1$ .

Now  $\mathbb{A}$  continues with the leaf returned by  $\mathbb{S}$ .

It is easy to check that the node  $b$  processed by the algorithm is always a leaf (as required). To see this, note that whenever the algorithm processes a node  $b$ , the sibling of every node on the path from the root to  $b$  is a leaf. (Hence actually the tree  $\mathcal{B}$  is always quite degenerate.)

Furthermore, using the Splitting Lemma, it is easy to show that the following invariants are maintained throughout the execution of the algorithm:

- (1) Let  $b$  be a node of the tree labelled  $(G_b, T_b)$ . Then the cycles labelling the edges on a path from the root to  $b$  are pairwise disjoint, and they are all disjoint from the graph  $G_b$ .
- (2) Let  $b$  be a node of the tree labelled  $(G_b, T_b)$ . Then  $T_b$  is a feedback vertex set of  $G_b$ .
- (3) Let  $b_1$  be a node labelled  $(G_1, T_1)$  of height  $i + 1$ , for some  $i \geq 0$ . Suppose that the parent  $b'$  is labelled  $(G', T')$  and that the sibling  $b_2$  of  $b$  is labelled  $(G_2, T_2)$ . Then  $G_1$  and  $G_2$  are vertex disjoint subgraphs of  $G$ . For all feedback vertex sets  $T'_1$  of  $G_1$  and  $T'_2$  of  $G_2$  the set  $T'_1 \cup T'_2 \cup (V(G) \setminus V(G_1 \cup G_2))$  is a feedback vertex set of  $G$  of size at most  $|T'_1| + |T'_2| + |T'| - (m_i + 1)$ . Furthermore, it holds that  $m_i < |T_1| + |T_2| \leq 2m_i$ .

For the lower bound  $m_i < |T_1| + |T_2|$  in (3) note that as soon as  $|T_1| + |T_2| \leq m_i$  for some children  $b_1, b_2$  of  $b'$  labelled  $(G_1, T_1)$  and  $(G_2, T_2)$  it holds that  $|T''| < |T'|$  for  $T'' := T'_1 \cup T'_2 \cup (V(G) \setminus V(G_1 \cup G_2))$  and the subroutine  $\mathbb{S}$  deletes  $b_1$  and  $b_2$ .

- (4) Suppose that at some point during the execution a leaf  $b_1$  labelled  $(G_1, T_1)$  is processed while the sibling  $b_2$  of  $b$  is labelled  $(G_2, T_2)$ . Then  $|T_1| \geq |T_2|$ .

(3) and (4) imply:

- (5) Suppose that at some point during the execution of  $\mathbb{A}$  a leaf  $b_1$  labelled  $(G_1, T_1)$  and of height  $i + 1$ , for some  $i \geq 0$ , is processed. Then  $m_i/2 \leq |T_1|$ .

Finally, note that throughout the execution:

- (6) The root is labelled  $(G, T_0)$  for some feedback vertex set  $T_0$  of size  $2m_0 \leq |T_0| = O(\tau^*(G) \cdot \log \tau^*(G) \cdot \log \log \tau^*(G))$ .

The first inequality in (6) holds as  $2m_0 = 2\mu(G) \leq \tau^*(G) \leq \tau(G)$  by the definition of  $\mu$ . Since  $m_{i+1} \leq m_i/4$ , it follows that throughout the execution:

- (7) Whenever a node  $b$  of height  $i$  labelled  $(G_b, T_b)$  is processed, we have  $2m_i \leq |T_b|$ .

This justifies the application of the Splitting Lemma in the algorithm.

Observe that (1) implies that whenever the algorithm halts, its output is a family of disjoint cycles. The size of this family is  $i^*$  if the algorithm halts at a leaf of height  $i^*$ , or  $i + \varphi(m_i)$ , if the algorithm halts at a leaf of height  $i < i^*$  in

a Linkage Step. As  $\chi(\mu(G)) = \chi(m_0) \leq \min\{i^*, i + \varphi(m_i)\}$  for  $0 \leq i < i^*$  and as  $\mu(G) = \zeta(\tau^*(G))$ , these families of cycles are sufficiently large.

It remains to prove that the execution always terminates within the desired running time bounds. To do this, we upperbound the number  $N$  of shrinking steps that might occur while building the tree. For the root of the tree at most  $|T| - \tau^*(G)$  shrinking steps can occur before we end up with a minimal feedback vertex set for the given graph  $G$ . For all nodes  $b'$  of height  $i \geq 1$  we claim that at most  $m_i \leq \mu(G)$  shrinking steps on children of  $b'$  can occur before the feedback vertex set for  $b'$  is shrunk and the children are deleted in subroutine  $\mathbb{S}$ . To see this, let  $b'$  be a node labelled  $(G', T')$  and of height  $i$  that has children  $b_1, b_2$  labelled  $(G_1, T_1)$  and  $(G_2, T_2)$ . Recall that by (3), the set  $T'' := T_1 \cup T_2 \cup (V(G) \setminus V(G_1 \cup G_2))$  is a feedback vertex set of  $G'$  of size at most  $|T_1| + |T_2| + |T'| - (m_i + 1)$ . Furthermore, it holds that  $|T_1| + |T_2| \leq 2m_i$ . So after at most  $m_i$  shrinking steps for the children of  $b$  it holds that  $b_1, b_2$  are labelled with  $(G_1, T_1)$  and  $(G_2, T_2)$  such that  $|T_1| + |T_2| \leq m_i$  and hence  $|T''| < |T'|$ . This implies that  $b'$  is relabelled by  $(G', T'')$  for this smaller feedback vertex set  $T''$  in subroutine  $\mathbb{S}$  after at most  $m_i$  shrinking steps on  $b_1, b_2$ .

As the tree  $\mathcal{B}$  always contains at most two nodes with height  $i$  for  $1 \leq i \leq i^*$  and as  $m_i \leq m_0 = \mu(G)$  for all  $1 \leq i \leq i^*$ , there occur at most  $(|T| - \tau^*(G)) \cdot \mu(G)^{i^*}$  shrinking steps in total. So by the definition of  $\mu(G)$  and  $i^*$  and as  $|T| = O(\tau^*(G) \cdot \log(\tau^*(G)) \cdot \log \log(\tau^*(G)))$ , the number  $N$  can be upper bounded by a function of  $\tau^*(G)$ . This implies that we can execute the algorithm by using the Splitting Lemma at most  $2i^* \cdot (N + 1)$  times and the Linkage Lemma at most once. In total, this results in a running time that is bounded by  $g(\tau^*(G)) \cdot |G|^{O(1)}$  for some computable function  $g$ . □

*Proof (Proof of Theorem 8).* The fpt approximability of the disjoint cycle problem follows from Lemma 11, Theorem 3, and Proposition 7 using the observation that the disjoint cycle problem is well-behaved for parameterized approximation. The polynomial running time can be obtained by Lemma 5. □

## 6 Concluding Remarks

We give an fpt approximation algorithm for the directed vertex disjoint cycle problem. We mainly see this result as a contribution to the evolving theory of parameterized approximability [3,4,6,14], as it provides the first natural example of a parameterized problem that is fpt approximable, but known to be hard to be solved exactly (W[1]-hard, to be precise).

Our algorithm is based on the connection between disjoint cycles and feedback vertex sets and a structure theory for directed graphs developed in this context by Reed et al. [16]. To establish our result, we had to make large parts of this structure theory algorithmic. This may turn out to be useful in other contexts. In particular, it may help to find an fpt algorithm for the directed feedback vertex set problem and hence solve one of the most prominent open problems in parameterized complexity theory.

## References

1. Alon, N.: Disjoint directed cycles. *Journal of Combinatorial Theory Series B* 68(2), 167–178 (1996)
2. Bang-Jensen, J., Gutin, G.: *Digraphs*. Springer, Heidelberg (2002)
3. Cai, L., Huang, X.: Fixed-parameter approximation: Conceptual framework and approximability results. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006*. LNCS, vol. 4169, pp. 96–108. Springer, Heidelberg (2006)
4. Chen, Y., Grohe, M., Grüber, M.: On parameterized approximability. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006*. LNCS, vol. 4169, pp. 109–120. Springer, Heidelberg (2006)
5. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer, Heidelberg (1999)
6. Downey, R.G., Fellows, M.R., McCartin, C.: Parameterized approximation algorithms. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006*. LNCS, vol. 4169, pp. 121–129. Springer, Heidelberg (2006)
7. Erdős, P., Pósa, L.: On the independent circuits contained in a graph. *Canadian Journal of Mathematics* 17, 347–352 (1965)
8. Even, G., Naor, J.S., Schieber, B., Sudan, M.: Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica* 20(2), 151–174 (1998)
9. Flum, J., Grohe, M.: *Parameterized Complexity Theory*. Springer, Heidelberg (2006)
10. Graham, R.L., Grötschel, M., Lovász, L. (eds.): *Handbook of Combinatorics (volume II, chapter Ramsey theory)*, pp. 1331–1403. Elsevier Science, Amsterdam (1995)
11. Grötschel, M., Lovasz, L., Schrijver, A.: *Geometric Algorithms and Combinatorial Optimization*, 2nd edn. Springer, Heidelberg (1993)
12. Guenin, B., Thomas, R.: Packing directed circuits exactly. To appear in *Combinatorica* (2006)
13. Gutin, G., Yeo, A.: Some parameterized problems on digraphs (Submitted 2006)
14. Marx, D.: Parameterized complexity and approximation algorithms. To appear in *The Computer Journal* (2006)
15. Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, Oxford (2006)
16. Reed, B., Robertson, N., Seymour, P., Thomas, R.: Packing directed circuits. *Combinatorica* 16(4), 535–554 (1996)
17. Reed, B., Shepherd, F.: The gallai-younger conjecture for planar graphs. *Combinatorica* 16(4), 555–566 (1996)
18. Salavatipour, M., Verstraete, J.: Disjoint cycles: Integrality gap, hardness, and approximation. In: Jünger, M., Kaibel, V. (eds.) *Integer Programming and Combinatorial Optimization*. LNCS, vol. 3509, pp. 51–65. Springer, Heidelberg (2005)
19. Seymour, P.: Packing directed circuits fractionally. *Combinatorica* 15(2), 281–288 (1995)
20. Slivkins, A.: Parameterized tractability of edge-disjoint paths on directed acyclic graphs. In: Di Battista, G., Zwick, U. (eds.) *ESA 2003*. LNCS, vol. 2832, pp. 482–493. Springer, Heidelberg (2003)
21. Younger, D.: Graphs with interlinked directed circuits. In: *Proceedings of the Midwest Symposium on Circuit Theory 2*, pages XVI 2.1–XVI 2.7 (1973)

# Linear Problem Kernels for NP-Hard Problems on Planar Graphs

Jiong Guo\* and Rolf Niedermeier

Institut für Informatik, Friedrich-Schiller-Universität Jena,  
Ernst-Abbe-Platz 2, D-07743 Jena, Germany  
{guo,niedermr}@minet.uni-jena.de

**Abstract.** We develop a generic framework for deriving linear-size problem kernels for NP-hard problems on planar graphs. We demonstrate the usefulness of our framework in several concrete case studies, giving new kernelization results for CONNECTED VERTEX COVER, MINIMUM EDGE DOMINATING SET, MAXIMUM TRIANGLE PACKING, and EFFICIENT DOMINATING SET on planar graphs. On the route to these results, we present effective, problem-specific data reduction rules that are useful in any approach attacking the computational intractability of these problems.

## 1 Introduction

Data reduction together with problem kernelization has been recognized as one of the primary contributions of parameterized complexity to practical algorithm design [9,15,21]. For instance, the NP-hard VERTEX COVER problem, where one asks for a set of at most  $k$  vertices such that all edges of a given graph have at least one endpoint in this set, has a problem kernel of  $2k$  vertices. That is, given a graph  $G$  and the parameter  $k$ , one can construct in polynomial time a graph  $G'$  consisting of only  $2k$  vertices and with a new parameter  $k' \leq k$  such that  $(G, k)$  is a yes-instance iff  $(G', k')$  is a yes-instance [20,8]. In particular, this means that VERTEX COVER can be efficiently preprocessed with a guaranteed quality of data reduction—the practical usefulness is confirmed by experimental work [1]. Note that a  $2k$ -vertex problem kernel is the best one may probably hope for because a  $(2 - \epsilon)k$ -vertex kernel with  $\epsilon > 0$  would imply a factor- $(2 - \epsilon)$  polynomial-time approximation algorithm for VERTEX COVER, solving a long standing open problem. Clearly, a  $k$ -vertex problem kernel for VERTEX COVER would imply P=NP. That is why so-called linear-size problem kernels (a linear function in the parameter  $k$ ) usually are considered as the “holy grail” in the field of kernelization and parameterized complexity analysis.

Unfortunately, so far there are not too many problems known with a problem kernel size as small as we have for VERTEX COVER. Moreover, strictly speaking, the  $2k$ -vertex problem kernel for VERTEX COVER is not really a linear-size problem kernel because the number of graph edges still may be  $O(k^2)$ . Apparently, the

---

\* Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.



situation changes when focussing attention on planar graphs where the number of vertices and the number of edges are linearly related. Although most NP-hard graph problems remain NP-hard when restricted to planar graphs, it has been observed that they behave much better in terms of approximability (see [5]) as well as in terms of fixed-parameter tractability (see [4]). In her seminal work, Baker [5] showed that a whole class of problems (including VERTEX COVER, INDEPENDENT SET, DOMINATING SET) possesses polynomial-time approximation schemes (PTAS), all derived from a general framework.

Concerning problem kernelization results on planar graphs where, in a sense, linear-size problem kernels can be seen as the parameterized counterpart of approximation schemes, so far only few isolated results are known [4,7,17,19]. In particular, it has been shown that the DOMINATING SET problem—which is  $W[2]$ -complete on general graphs, meaning that there is no hope for a problem kernel at all [9,21]—has a linear-size problem kernel when restricted to planar graphs [4]. This result goes along with the development of simple but effective data reduction rules whose practical usefulness has been empirically confirmed [2,4]. In this work, “in the spirit of Baker”, we develop a general framework that allows for a systematic approach to derive linear-size problem kernels for planar graph problems. In particular, our methodology offers concrete starting-points for developing effective data reduction rules, the central part of any form of problem kernelization. Inspired by the work of Alber et al. [4], which focuses on the DOMINATING SET problem, we show what the common features are that lie at the heart of linear-size problem kernels for planar graph problems. In particular, we provide a concrete route of attack which serves as a tool for developing data reduction rules. Doing so, we provide a number of case studies together with new results, including the problems CONNECTED VERTEX COVER, EDGE DOMINATING SET, MAXIMUM TRIANGLE PACKING, and EFFICIENT DOMINATING SET, all of which are shown to have linear-size problem kernels with concrete upper bounds. Note that, although based on the general framework, all corresponding data reduction rules that had to be newly developed are—of course—problem-specific. The development of these rules still needs novel ideas in each specific case and is far from being routine. Still, our framework offers a guiding star to find them.

Most proofs are deferred to the full version of this paper.

## 2 Preliminaries

Parameterized algorithmics is a two-dimensional framework for studying the computational complexity of problems [9,21]. A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction rules*, often yielding a *reduction to a problem kernel (kernelization)*. Herein, the goal is, given any problem instance  $x$  with parameter  $k$ , to transform it in polynomial time into a new instance  $x'$  with parameter  $k'$  such that the size of  $x'$  is

---

<sup>1</sup> Indeed, the data reduction rules can be applied to all sorts of graphs and not only to planar ones—the rules are particularly effective for sparse graphs.

bounded from above by some function only depending on  $k$ ,  $k' \leq k$ , and  $(x, k)$  is a yes-instance iff  $(x', k')$  is a yes-instance. Then, the problem kernel is called to be *linear* if  $|x'| = O(k)$ . This transformation is accomplished by applying data reduction rules. A data reduction rule is *correct* if the new instance after an application of this rule is a yes-instance iff the original instance is a yes-instance. Throughout this paper, we call a problem instance *reduced* if the corresponding data reduction rules cannot be applied any more.

We only consider *undirected* graphs  $G = (V, E)$ , where  $V$  is the set of vertices and  $E$  is the set of edges. We use  $n$  to denote the number of vertices and  $m$  to denote the number of edges of a given graph. The *neighborhood*  $N(v)$  of a vertex  $v \in V$  is the set of vertices that are adjacent to  $v$ . The *degree* of a vertex  $v$  is the size of  $N(v)$ . We use  $N[v]$  to denote the *closed* neighborhood of  $v$ , that is,  $N[v] := N(v) \cup \{v\}$ . For a set of vertices  $V' \subseteq V$ , the *induced subgraph*  $G[V']$  is the graph with the vertex set  $V'$  and the edge set  $\{\{v, w\} \in E \mid v, w \in V'\}$ . A subset  $I$  of vertices is called an *independent set* if  $G[I]$  has no edge. We implicitly assume that all paths that we deal with here are *simple*, that is, every vertex is contained at most once in a path. The length of a path is defined as the number of edges used by the path. The distance  $d(u, v)$  between two vertices  $u, v$  is the length of a shortest path between  $u, v$ . The distance  $d(e, w)$  between an edge  $e = \{u, v\}$  and a vertex  $w$  is the minimum of  $d(u, w)$  and  $d(v, w)$ . If a graph can be drawn in the plane without edge crossings then it is called a *planar graph*. A *plane graph* is a planar graph with a fixed embedding in the plane. Throughout this paper, we assume that we are working with an arbitrary but fixed embedding of  $G$  in the plane; whenever this embedding is of relevance, we refer to  $G$  as being *plane* instead of planar.

### 3 General Framework

In this section, we describe a general framework for systematically deriving linear problem kernels for NP-hard problems on planar graphs. Although in this (single) case not improving on previous results, for reason of simplicity, we use VERTEX COVER as a running example. The problem is, given a graph  $G = (V, E)$  and  $k \geq 0$ , to find a subset  $C \subseteq V$  of at most  $k$  vertices such that every edge has at least one endpoint in  $C$ . The remainder of this section is structured by exhibiting the four basic components of our methodology.

*Component 1: Problem-specific distance property.* The problems amenable to our framework have to admit a *distance property* defined as follows:

**Definition 1.** *A graph problem on input  $G = (V, E)$  is said to admit a distance property with constants  $c_V$  and  $c_E$  if, for every solution set  $S$  with the vertex set  $V(S)$ , it holds that, for every vertex  $u \in V$ , there exists a vertex  $v \in V(S)$  with  $d(u, v) \leq c_V$ , and, for every edge  $e \in E$ , there exists a vertex  $v \in V(S)$  with  $d(e, v) \leq c_E$ .*

Note that  $c_V - 1 \leq c_E \leq c_V$ . The distance property is the only prerequisite for applying our framework to a specific graph problem.

**Example:** The distance property is valid for VERTEX COVER with  $c_V = 1$  and  $c_E = 0$ , since every edge of  $E$  has to be incident to a covering vertex.

*Component 2: Region decomposition.* We divide the vertices not in  $V(S)$  into two categories based on whether they lie in the vicinity of either at least two vertices of  $V(S)$  or only one vertex of  $V(S)$ . The former vertices will build so-called *regions* leading to a decomposition of the planar graph.

**Definition 2.** A region  $R(u, v)$  between two distinct vertices  $u, v \in V(S)$  is a closed subset of the plane with the following properties:

1. The boundary of  $R(u, v)$  is formed by two length-at-most- $(c_V + c_E + 1)$  paths between  $u$  and  $v$ . (These two paths do not need to be disjoint or simple.)
2. All vertices which lie on the boundary or strictly inside of the region  $R(u, v)$  have distance at most  $c_V$  to at least one of the vertices  $u$  and  $v$  and all edges whose both endpoints lie on the boundary or strictly inside of the region  $R(u, v)$  have distance at most  $c_E$  to at least one of the vertices  $u$  and  $v$ .
3. With the exception of  $u$  and  $v$ , none of the vertices which lie inside of the region  $R(u, v)$  are from  $V(S)$ .

The vertices  $u$  and  $v$  are called the anchor vertices of  $R(u, v)$ . A vertex is said to lie inside of  $R(u, v)$  if it is either a boundary vertex of  $R(u, v)$  or if it lies strictly inside of  $R(u, v)$ . We use  $V(R(u, v))$  to denote the set of vertices that lie inside of a region  $R(u, v)$ .

Using Definition 2, the graph can be partitioned by a so-called *region decomposition*.

**Definition 3.** An  $S$ -region decomposition of a graph is a set  $\mathcal{R}$  of regions such that there is no vertex that lies strictly inside of more than one region from  $\mathcal{R}$  (the boundaries of regions may touch each other, however).

For an  $S$ -region decomposition  $\mathcal{R}$ , let  $V(\mathcal{R}) := \bigcup_{R \in \mathcal{R}} V(R)$ . An  $S$ -region decomposition  $\mathcal{R}$  is called maximal if there is no region  $R \notin \mathcal{R}$  such that  $\mathcal{R}' := \mathcal{R} \cup \{R\}$  is an  $S$ -region decomposition with  $V(\mathcal{R}) \subsetneq V(\mathcal{R}')$ .

As a basis for linear kernelization results, our framework makes use of the fact that the number of regions in a maximal region decomposition  $\mathcal{R}$  for a given solution  $S$  can be upper-bounded by  $c_V \cdot (3|V(S)| - 6)$ . This generalizes a result of Alber et al. [4].

**Lemma 1.** Let  $P$  be a graph problem admitting the distance property with  $c_V$  and  $c_E$  and let  $S$  be a solution of  $P$  on a plane graph  $G = (V, E)$ . Then, there is a maximal  $S$ -region decomposition  $\mathcal{R}$  for the input graph  $G$  that consists of at most  $c_V \cdot (3|V(S)| - 6)$  regions.

**Example:** Since VERTEX COVER admits the distance property with  $c_V = 1$  and  $c_E = 0$ , the maximal region decomposition consists of regions with boundary paths of length at most two. By Lemma 1, we know that for a VERTEX COVER solution  $C$  of size at most  $k$  we have at most  $3k - 6$  regions in a maximal region decomposition.

*Component 3: Local neighborhoods for data reduction rule design.* The core algorithmic part of our methodology is based on the following two definitions that serve for developing data reduction rules yielding linear-size problem kernels.

**Definition 4.** *Given a problem admitting the distance property with constants  $c_V$  and  $c_E$ , the private neighborhood  $N_p(u)$  of a vertex  $u$  consists of the vertices that have distance at most  $c_V$  to  $u$ , that are not adjacent to the vertices with distance at least  $c_V + 1$  to  $u$ , and that are not incident to any edge with distance more than  $c_E$  to  $u$ .*

**Example:** For VERTEX COVER, the private neighborhood  $N_p(u)$  of a vertex  $u$  consists only of the degree-one vertices from  $N(u)$ . Therefore, the corresponding private neighborhood rule deals with degree-one vertices:

**Private neighborhood rule for VERTEX COVER:** If  $N_p(u) \neq \emptyset$ , then add  $u$  to  $C$  and remove  $N_p(u)$  from the graph. Decrease the parameter  $k$  by one.

**Definition 5.** *Given a problem admitting the distance property with constants  $c_V$  and  $c_E$ , the joint private neighborhood  $N_p(u, v)$  of two vertices  $u, v \in V$  consists of the vertices that have distance at most  $c_V$  to  $u$  or  $v$ , that are not adjacent to the vertices which have distance at least  $c_V + 1$  to both  $u$  and  $v$ , and that are not incident to any edge with distance more than  $c_E$  to both  $u$  and  $v$ .*

**Example:** In VERTEX COVER, the joint private neighborhood of  $u$  and  $v$  consists of their common neighbors and their degree-one neighbors. Since the private neighborhood rule deals with the degree-one neighbors, for the corresponding data reduction rule we only consider the common neighbors of  $u$  and  $v$ :

**Joint private neighborhood rule for VERTEX COVER:** If two vertices  $u, v$  have at least two common degree-two neighbors, then add  $u$  and  $v$  to  $C$  and remove  $u, v$  and their common degree-two neighbors from the graph. Decrease the parameter  $k$  by two.

Generally speaking, if  $N_p(v)$  of a vertex  $v$  (or  $N_p(u, v)$  of vertices  $u$  and  $v$ ) contains too many vertices, then any solution of a minimization problem has to include  $v$  (or at least one of  $u$  and  $v$ ). For maximization problems, we can conclude that including only  $v$  from  $N_p(v)$  (or only  $u$  and  $v$  from  $N_p(u, v)$ ) cannot lead to a solution. This provides a useful argument for showing upper bounds on the region sizes in the problem kernel size analysis.

*Component 4: Mathematical analysis of problem kernel size.* Having derived problem-specific data reduction rules, the next step in our method is to prove that there are only constantly many vertices inside of a region. Together with Lemma II, this implies the upper bound  $O(|V(S)|)$  for all vertices inside of all regions of the reduced graph.

**Example:** For VERTEX COVER, the constant size for every region follows almost directly from the given two rules:

**Lemma 2.** *Given a vertex cover  $C$  of a reduced planar graph  $G = (V, E)$ , every region of a maximal region decomposition contains at most three vertices which are not from  $C$ .*

*Proof.* Consider a maximal region decomposition and let  $R$  denote an arbitrary region with boundary paths of length at most two. Let  $u, v$  be the two vertices in  $V(R)$  that are from  $C$ . Clearly, at most two vertices on the boundary paths can be from  $V \setminus C$ . We claim that there is at most one vertex lying strictly inside of  $R$ . To show this, suppose that there are two vertices  $x, y$  strictly inside of  $R$ . Then, by the definition of regions,  $x, y \in N(u) \cup N(v)$ . Due to the private neighborhood rule, each of  $x$  and  $y$  has at least two neighbors in  $V(R) \cap C$ . Since  $u$  and  $v$  are the only  $C$ -vertices in  $R$ , the vertices  $x, y$  are common neighbors of  $u$  and  $v$  and have degree two. This implies that the joint private neighborhood rule can be applied, a contradiction to the fact that  $G$  is reduced. Therefore, at most one vertex lies strictly inside of  $R$  and the lemma follows.  $\square$

Note that, if, in addition to the above two rules, the “folding” rule introduced by Chen et al. [8] is applied, one can show that every region contains at most two vertices from  $V \setminus C$ .

To complete the proof for a linear-size problem kernel, our method requires to upper-bound the number of vertices not contained in any region. To do so, the private neighborhood rule is crucial.

**Example:** In the case of VERTEX COVER, the private neighborhood rule guarantees that there is no vertex lying outside of the regions of a maximal region decomposition:

**Lemma 3.** *Let  $\mathcal{R}$  be a maximal region decomposition of a reduced planar graph for VERTEX COVER. Then, there is no vertex lying outside of the regions in  $\mathcal{R}$ .*

*Proof.* Suppose that there is such a vertex  $x$ . It cannot be a degree-one vertex and it cannot be adjacent to a vertex not from  $C$ . Thus,  $N(x) \subseteq C$ . Then, we can arbitrarily pick two from  $x$ ’s neighbors and have a region  $R$  that is a path consisting of  $x$  and these two neighbors. By adding  $R$  to  $\mathcal{R}$  we get a new region decomposition  $\mathcal{R}'$  with  $V(\mathcal{R}) \subsetneq V(\mathcal{R}')$ , a contradiction to the fact that  $\mathcal{R}$  is maximal.  $\square$

Finally, to give an overall kernel size bound, we only need to add up the number of vertices inside of regions and the number of vertices outside of regions.

**Example:** With the two upper bounds given in Lemmas [2] and [3], we arrive at our linear kernelization result for VERTEX COVER on planar graphs:

**Proposition 1.** *VERTEX COVER on planar graphs admits a  $10k$ -vertex problem kernel.*

*Proof.* By Lemma [1], there are at most  $3k - 6$  regions in a maximal region decomposition. Together with Lemma [2], there can be at most  $9k - 18$  vertices from  $V \setminus C$  lying inside of regions. By Lemma [3], no vertex can be outside of regions. Thus, altogether, we have  $10k - 18$  vertices in the reduced graph.  $\square$

## 4 Case Studies

Now, we exhibit the versatility of our general methodology.

*Connected Vertex Cover.* Given a graph  $G = (V, E)$  and a non-negative integer  $k$ , the CONNECTED VERTEX COVER problem asks for a set  $C$  of at most  $k$  vertices such that  $G[C]$  is connected and  $C$  is a vertex cover of  $G$ . This problem is NP-complete on planar graphs [13]. Until now only an exponential-size kernel in general graphs is known [16].

Since a connected vertex cover is also a vertex cover, the distance property holds for this problem with  $c_V = 1$  and  $c_E = 0$ . Thus, the regions in a maximal region decomposition for CONNECTED VERTEX COVER have also boundary paths of length at most two and we have at most  $3k - 6$  regions in a maximal region decomposition. Moreover, the private neighborhood and the joint private neighborhood are defined in the same way as for VERTEX COVER.

The data reduction rules are similar to the ones for VERTEX COVER. However, to guarantee the resulting vertex cover being connected, we use gadgets:

**Private neighborhood rule:** If a vertex has more than one degree-one neighbor, then except for one remove all of these neighbors.

**Joint private neighborhood rule:** If two vertices have more than two common degree-two neighbors, then remove all of these neighbors except for two.

**Theorem 1.** *CONNECTED VERTEX COVER on planar graphs admits a  $14k$ -vertex problem kernel.*

*Proof.* First, consider a region  $R$  in a maximal  $C$ -region decomposition  $\mathcal{R}$  for a connected vertex cover  $C$  with  $|C| \leq k$ . Note that strictly inside of a region there cannot be vertices of degree more than two because this would imply uncovered edges. Due to the joint private neighborhood rule, there can be at most two degree-two vertices lying strictly inside of  $R$ . Since there can be at most two vertices from  $V \setminus C$  lying on the boundary of a region, each region can contain at most four vertices from  $V \setminus C$ . Since the graph is reduced with respect to the private neighborhood rule, each vertex in  $C$  can have at most one degree-one neighbor not lying in a region of  $\mathcal{R}$ . Therefore, we altogether have at most  $4 \cdot (3k - 6)$  vertices from  $V \setminus C$  which lie inside of regions and at most  $k$  vertices outside of regions. Together with  $|C| \leq k$ , the size bound follows.  $\square$

*Edge Domination Set.* Given a graph  $G = (V, E)$  and a non-negative integer  $k$ , the EDGE DOMINATING SET problem asks for a set  $E'$  of at most  $k$  edges such that all edges in  $E$  share at least one endpoint with some edge in  $E'$ . This problem is NP-complete on planar graphs [14]. Based on its equivalence to the MINIMUM MAXIMAL MATCHING problem, a problem kernel with  $O(k^2)$  vertices for general graphs has been derived [22]. It is easy to observe that EDGE DOMINATING SET has the same distance parameters as VERTEX COVER. Therefore, the same two data reduction rules for CONNECTED VERTEX COVER apply.

**Theorem 2.** EDGE DOMINATING SET on planar graphs admits a  $14k$ -vertex problem kernel.

*Maximum Triangle Packing.* Given a graph  $G = (V, E)$  and a non-negative integer  $k$ , the MAXIMUM TRIANGLE PACKING problem asks for a set  $P$  of at least  $k$  vertex-disjoint triangles in  $G$ . The set  $P$  is called a *triangle packing* of  $G$ . This problem is NP-complete on planar graphs [14]. A problem kernel with  $O(k^3)$  vertices is known for general graphs [10].

At first glance, MAXIMUM TRIANGLE PACKING does not admit the required distance property. However, the following data reduction rule can be applied.

**Cleaning rule:** Remove all vertices and edges that are not in a triangle.

In an instance where the cleaning rule does not apply, every vertex and every edge has a distance at most  $c_V = 1$  and  $c_E = 1$ , respectively, to some vertex occurring in a triangle packing that cannot be extended by a triangle. Then, the regions in a maximal region decomposition for MAXIMUM TRIANGLE PACKING have boundary paths of length at most three.

Consider the private neighborhood  $N_p(u)$  of a vertex  $u$ . According to Definition 4, all vertices  $v \in N_p(u)$  have to satisfy  $N[v] \subseteq N[u]$ . We apply the following data reduction rule dealing with private neighborhoods.

**Private neighborhood rule:** If a vertex  $u$  has two neighbors  $v, w$  that form a triangle with  $u$  but are not involved in any other triangles that do not contain  $u$ , then remove  $u, v, w$  and decrease the parameter  $k$  by one.

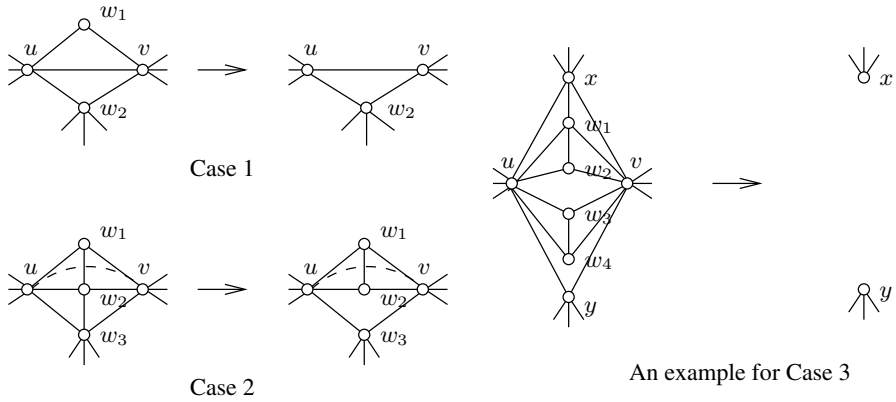
Next, we consider the joint private neighborhood  $N_p(u, v)$  of two vertices  $u, v$ . According to Definition 5, every vertex  $x \in N_p(u, v)$  has to satisfy  $N[x] \subseteq N[u] \cup N[v]$ .

**Joint private neighborhood rule:** If two vertices  $u, v$  have more than two common neighbors, then consider the following cases.

- Case 1: If  $u$  and  $v$  have two common neighbors  $w_1$  and  $w_2$  such that  $w_1$  has degree two and  $w_2$  is only contained in triangles that also contain  $u$  or  $v$ , then remove  $w_1$ .
- Case 2: If  $u$  and  $v$  have three common neighbors  $w_1, w_2, w_3$  such that  $N(w_1) = \{u, v, w_2\}$  and  $N(w_2) = \{u, v, w_1, w_3\}$ , then remove edge  $\{w_2, w_3\}$ .
- Case 3: If there are four vertices  $w_1, w_2, w_3, w_4$  such that  $u, w_1, w_2$  form a triangle,  $v, w_3, w_4$  form another one, and there is no other triangle that contains one of  $w_1, w_2, w_3, w_4$  but none of  $u, v$ , then remove  $u, v, w_1, w_2, w_3, w_4$  and decrease the parameter  $k$  by two.

Note that, after the application of the cleaning rule, every vertex has to be in a triangle. Therefore, in Case 1, there has to be an edge between  $u$  and  $v$ . The three cases of the joint private neighborhood rule are illustrated in Fig. 1.

In order to give a linear-size problem kernel, we need only the first and the third case of the joint private neighborhood rule. However, including the second case allows us to give a better bound on the maximum size of regions in a maximal region decomposition of the reduced graphs as stated in the following, allowing for a smaller upper bound on the problem kernel size.



**Fig. 1.** Illustration of the three cases of the joint private neighborhood rule for MAXIMUM TRIANGLE PACKING. A dashed line means a possibly existing edge.

To prove that there is only a constant number of vertices inside of each region, graph structures that we call *diamonds*<sup>2</sup> are of great importance.

**Definition 6.** Let  $u$  and  $v$  be two vertices in a plane graph  $G$ . A diamond  $D(u, v)$  is a closed area of the plane that is bounded by two length-2 paths between  $u$  and  $v$  such that every vertex that lies inside this area is a neighbor of both  $u$  and  $v$ . If  $i$  vertices lie strictly inside a diamond, then it is said to have  $(i + 1)$  facets.

**Lemma 4.** In a reduced planar graph, a diamond can have at most five facets.

Now, we state upper bounds on the number of vertices in a region and on the number of vertices outside of all regions.

**Lemma 5.** Consider a planar graph  $G = (V, E)$  for which any triangle packing contains at most  $k$  triangles. If  $G$  is reduced, then, in a maximal region decomposition of  $G$ ,

1. every region can contain at most 71 vertices, and
2. there are less than  $108k$  vertices lying outside of regions.

**Theorem 3.** MAXIMUM TRIANGLE PACKING on planar graphs admits a  $732k$ -vertex problem kernel.

*Efficient Dominating Set.* Given a graph  $G = (V, E)$  and a non-negative integer  $k$ , the EFFICIENT DOMINATING SET problem is to decide whether there exists an independent set  $I$  such that every vertex in  $V \setminus I$  has exactly one neighbor in  $I$ . A solution set of this problem is called an *efficient dominating set*. This

<sup>2</sup> Note that standard graph theory uses the term “diamond” to denote a 4-cycle with exactly one chord. We abuse this term here for obvious reasons. We remark that diamonds also played a decisive role in proving a linear-size problem kernel for DOMINATING SET on planar graphs [4].



problem is NP-complete on planar graphs of maximum degree three [11]. In the literature, EFFICIENT DOMINATING SET also appears under the names PERFECT CODE, INDEPENDENT PERFECT DOMINATING SET, and PERFECT DOMINATING SET. Lu and Tang [18] provided an overview of complexity results for EFFICIENT DOMINATING SET.

Bange et al. [6] showed that if a graph  $G$  has an efficient dominating set, then all efficient dominating sets of  $G$  have the same cardinality, and this is the same as the domination number of  $G$ , where the domination number is the cardinality of a minimum dominating set of  $G$ . Hence, the parameterized version of EFFICIENT DOMINATING SET additionally has a non-negative integer  $k$  as input and asks for an efficient dominating set of size *exactly*  $k$ . EFFICIENT DOMINATING SET is W[1]-hard in general graphs [9]. To our knowledge, there is no kernelization result known for this problem on planar graphs. Note that the linear-size problem kernel for DOMINATING SET on planar graphs does not imply a linear-size problem kernel for EFFICIENT DOMINATING SET on planar graphs since the data reduction rules applied by Alber et al. [4] for deriving the problem kernel apparently do not work for EFFICIENT DOMINATING SET.

Since every efficient dominating set also is a dominating set, the distance property holds for EFFICIENT DOMINATING SET with  $c_V = 1$  and  $c_E = 1$ . The boundary paths of the regions in a maximal region decomposition have length at most three. Note that EFFICIENT DOMINATING SET has the same distance parameters  $c_V$  and  $c_E$  as MAXIMUM TRIANGLE PACKING. Therefore, the private neighborhood and the joint private neighborhood are the same in both cases.

In the following we describe two data reduction rules for EFFICIENT DOMINATING SET. Actually, the reduction rules apply to a more general setting where we are additionally given a subset of vertices  $F \subseteq V$  which may not be added to the efficient dominating set. In the following rules, whenever we would be forced to add a vertex in  $F$  to a solution set  $I$ , we report that the given instance has no efficient dominating set.

**Private neighborhood rule:** Consider the following two cases for a vertex  $v$  with  $N_p(u) \neq \emptyset$ :

- Case 1. If there is no vertex  $v \in N(u)$  such that  $v$  dominates all vertices in  $N_p(u)$ , then add  $u$  to the efficient dominating set  $I$ , remove all vertices in  $N[u]$  from the graph, add to  $F$  all vertices which are not in  $N[u]$  but adjacent to some vertex in  $N(u)$ , and decrease the parameter  $k$  by one.
- Case 2. If there is exactly one vertex  $v \in N(u)$  dominating all vertices in  $N_p(u)$ , then remove all vertices in  $N_p(u) \setminus \{v\}$  and add two new non-adjacent vertices  $x, y$  and connect them to both  $u$  and  $v$ .

**Joint private neighborhood rule:** Consider the following two cases for two vertices  $u, v$  with  $N_p(u, v) \neq \emptyset$ :

- Case 1. If  $u, v$  have two common neighbors  $x, y$  such that  $\{x, y\} \notin E$ ,  $N(x) \subsetneq N(u) \cap N(v)$ ,  $N(y) \subsetneq N(u) \cap N(v)$ , and  $N(x) \cap N(y) = \{u, v\}$ , then remove  $(N(u) \cap N(v)) \setminus \{x, y\}$  and add those vertices to  $F$  that are not in  $N(u) \cap N(v)$  but adjacent to some vertex in  $N(u) \cap N(v)$ .

- Case 2. Enumerate all subsets of  $N[u, v] := N[u] \cup N[v]$  that induce independent sets of size at most two and whose vertices are adjacent to all vertices in  $N_p(u, v)$ . If there is a vertex  $w$  occurring in all of these sets, then add  $w$  to the efficient dominating set, remove  $N[w]$  from the graph, add to  $F$  all vertices which are not in  $N[w]$  but adjacent to some vertex in  $N(w)$ , and decrease the parameter  $k$  by one.

**Lemma 6.** (1) *In a reduced planar graph, a diamond can have at most four facets.*

(2) *In a maximal region decomposition of a reduced planar graph, every region contains at most 28 vertices.*

(3) *In a maximal region decomposition of a reduced planar graph, there are at most  $5k$  vertices lying outside of regions.*

**Theorem 4.** EFFICIENT DOMINATING SET on planar graphs admits a  $84k$ -vertex problem kernel.

## 5 Outlook

There are numerous avenues for future research. First, it is promising to look into further improving the constant factors of our kernel bounds, similarly as Chen et al. [7] did for DOMINATING SET [4]. Second, again referring to Chen et al. [7] and the lower bounds on (linear) kernel sizes derived there for VERTEX COVER and DOMINATING SET on planar graphs, based on our framework, similar lower bound investigations may now be undertaken for other problems. Third, it appears natural to further extend the list of concrete problem kernel bounds for problems we did not touch here—further domination problems being obvious candidates. Observe that we can extend the list of linear problem kernel results using other further problems studied by Baker [5]. Fourth, as the linear-size problem kernel results for DOMINATING SET on planar graphs have been extended to graphs of bounded genus [12], it is tempting to generalize our whole framework in the same style. Fifth, for DOMINATING SET, a generic set of data reduction rules has been designed [3]—analogous studies now may be fruitful for all problems fitting into our framework.

## References

1. Abu-Khzam, F.N., Collins, R.L., Fellows, M.R., Langston, M.A., Suters, W.H., Symons, C.T.: Kernelization algorithms for the Vertex Cover problem: theory and experiments. In: Proc. 6th ACM-SIAM ALLENEX, pp. 62–69. ACM Press, New York (2004)
2. Alber, J., Betzler, N., Niedermeier, R.: Experiments on data reduction for optimal domination in networks. *Annals of Operations Research* 146(1), 105–117 (2006)
3. Alber, J., Dorn, B., Niedermeier, R.: A general data reduction scheme for domination in graphs. In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2006. LNCS, vol. 3831, pp. 137–147. Springer, Heidelberg (2006)

4. Alber, J., Fellows, M.R., Niedermeier, R.: Polynomial time data reduction for Dominating Set. *Journal of the ACM* 51(3), 363–384 (2004)
5. Baker, B.S.: Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM* 41(1), 153–180 (1994)
6. Bange, D.W., Barkauskas, A.E., Slater, P.J.: Efficient dominating sets in graphs. In: *Proc. 3rd Conference on Discrete Mathematics*. SIAM, pp. 189–199 (1988) **SIAM**
7. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. In: Diekert, V., Durand, B. (eds.) *STACS 2005*. LNCS, vol. 3404, pp. 269–280. Springer, Heidelberg (2005)
8. Chen, J., Kanj, I.A., Jia, W.: Vertex Cover: Further observations and further improvements. *Journal of Algorithms* 41, 280–301 (2001)
9. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer, Heidelberg (1999)
10. Fellows, M.R., Heggernes, P., Rosamond, F.A., Sloper, C., Telle, J.A.: Finding  $k$  disjoint triangles in an arbitrary graph. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) *WG 2004*. LNCS, vol. 3353, pp. 235–244. Springer, Heidelberg (2004)
11. Fellows, M.R., Hoover, M.: Perfect domination. *Australian Journal of Combinatorics* 3, 141–150 (1991)
12. Fomin, F.V., Thilikos, D.M.: Fast parameterized algorithms for graphs on surfaces: Linear kernel and exponential speed-up. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) *ICALP 2004*. LNCS, vol. 3142, pp. 581–592. Springer, Heidelberg (2004)
13. Garey, M.R., Johnson, D.S.: The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics* 32, 826–834 (1977)
14. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York (1979)
15. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. *SIGACT News* 38(1), 31–45 (2007)
16. Guo, J., Niedermeier, R., Wernicke, S.: Parameterized complexity of generalized Vertex Cover problems. In: Dehne, F., López-Ortiz, A., Sack, J.-R. (eds.) *WADS 2005*. LNCS, vol. 3608, pp. 36–48. Springer, Heidelberg (2005) Long version to appear under the title Parameterized complexity of Vertex Cover variants in *Theory of Computing Systems*
17. Guo, J., Niedermeier, R., Wernicke, S.: Fixed-parameter tractability results for full-degree spanning tree and its dual. In: Fischer, K., Timm, I.J., André, E., Zhong, N. (eds.) *MATES 2006*. LNCS (LNAI), vol. 4196, pp. 203–214. Springer, Heidelberg (2006)
18. Lu, C.L., Tang, C.Y.: Weighted efficient domination problem on some perfect graphs. *Discrete Applied Mathematics* 117, 163–182 (2002)
19. Moser, H., Sikdar, S.: The parameterized complexity of the induced matching problem in planar graphs. In: *Proc. 1st International Frontiers of Algorithmics Workshop (FAW 2007)*. LNCS, Springer, Heidelberg (2007)
20. Nemhauser, G.L., Trotter, L.E.: Vertex packing: structural properties and algorithms. *Mathematical Programming* 8, 232–248 (1975)
21. Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, Oxford (2006)
22. Prieto, E.: *Systematic Kernelization in FPT Algorithm Design*. PhD thesis, Department of Computer Science, University of Newcastle, Australia (2005)

# Private Locally Decodable Codes\*

Rafail Ostrovsky, Omkant Pandey, and Amit Sahai

Department of Computer Science  
University of California, Los Angeles 90095  
{rafail,omkant,sahai}@cs.ucla.edu

**Abstract.** We consider the problem of constructing efficient locally decodable codes in the presence of a computationally bounded adversary. Assuming the existence of one-way functions, we construct *efficient* locally decodable codes with positive information rate and *low* (almost optimal) query complexity which can correctly decode any given bit of the message from constant channel error rate  $\rho$ . This compares favorably to our state of knowledge locally-decodable codes without cryptographic assumptions. For all our constructions, the probability for any polynomial-time adversary, that the decoding algorithm incorrectly decodes any bit of the message is negligible in the security parameter.

## 1 Introduction

When a *message*  $\mathbf{x}$  is sent over a *channel*  $\mathcal{C}$ , the channel might introduce some *errors* so that the received message differs from the original message  $\mathbf{x}$ . To deal with this, the *sender* typically encodes the given message to obtain a *codeword*  $\mathbf{y}$  so that  $\mathbf{x}$  can be recovered even if the received codeword  $\mathbf{y}'$  differs from the original encoding  $\mathbf{y}$  in some of the places.

The message is represented by a sequence of  $k$  symbols from alphabet  $\Sigma$ . The codeword is also represented as a sequence of  $K$  symbols from the same alphabet  $\Sigma$ . The encoding function is denoted by  $\mathcal{S} : \Sigma^k \rightarrow \Sigma^K$  and the decoding function is denoted by  $\mathcal{R} : \Sigma^K \rightarrow \Sigma^k$ . The *information rate* (or simply *rate*) of the code is  $k/K$  and measures the amount of extra information needed by the code for correctly decoding from errors. Such a coding scheme is called a  $(K, k)_q$ -coding scheme, where  $q = |\Sigma|$ .

When the whole message  $\mathbf{x}$  should be recovered from the corrupted codeword  $\mathbf{y}'$ , the decoding algorithm reads  $\mathbf{y}'$  entirely. If one is interested in reading only one bit of  $\mathbf{x}$ , more efficient coding schemes are possible. In particular, it is possible

---

\* Part of this work was done when all the authors were at IPAM. The first author is supported in part by NSF Cybertrust grant No. 0430254, Xerox Innovation group Award and IBM Faculty Award. The second and third authors were supported in part from grants from the NSF ITR and Cybertrust programs (including grants 0627781, 0456717, and 0205594), a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, and an Alfred P. Sloan Foundation Research Fellowship.

to construct codes which can decode a single bit of  $\mathbf{x}$  by reading only a few bits of  $\mathbf{y}'$ . Such codes are called locally decodable codes (LDCs) [1,22,11].

Informally, a locally decodable code with *query complexity*  $\ell$ , *error rate*  $\rho$ , and *error correction probability*  $p$  is a pair of algorithms  $(\mathcal{S}, \mathcal{R})$ , where  $\mathcal{S}$  is the encoding algorithm and  $\mathcal{R}$  is the decoding algorithm, such that the decoding algorithm makes at most  $\ell$  queries into the corrupted codeword  $\mathbf{y}'$  and recovers any given bit  $j$  of  $\mathbf{x}$  with probability  $p$  or more if  $\mathbf{y}'$  differs from  $\mathbf{y}$  in at most a  $\rho$  fraction of alphabets. For brevity, such a code is sometimes written as  $(\ell, \rho, p)$ -LDC and we require that  $p > 0.5$ . An LDC is called *adaptive* if the queries of  $\mathcal{R}$  depend upon the answers of previous queries. It is called *non-adaptive* if they depend only on the random coins of  $\mathcal{R}$ .

Of course, locally decodable codes with high information rate, high error rate, high error correction probability, and low query complexity are most desirable. Low alphabet sizes ( $q = |\Sigma|$ ) are desirable too as most channels are best at transmitting only bits.

Locally decodable codes have found several notable applications. In complexity theory [6], PCPs [1], and so on. In cryptography they have been useful due to their interesting connection with private information retrieval protocols [4,13,2]. Their interesting properties make them applicable in several database applications such as fault-tolerant data storage [11]. It is tempting to say that constructions of good locally decodable codes can yield benefits to several related fields of computer science.

**MODELING THE NOISY CHANNEL.** The nature of channel errors plays an important role in the design of good error correcting codes. Historically, there are two popular ways of modeling a noisy channel: Shannon's model and Hamming's model. In Shannon's *symmetric channel* model, each symbol is changed to a random different one independently with some fixed probability. In Hamming's *adversarial channel* model, symbols get changed in the worst possible manner subject to an upper bound on the number of errors (such as a constant fraction of the size of the codeword). It should be noted that Hamming's channels are computationally *unbounded*. As a consequence, good error-correcting codes in Hamming's model ensure robustness of the coding scheme. But at the same time, constructing error correcting codes becomes more challenging in this model. In particular, *good*<sup>1</sup> locally decodable codes are not known to exist in this model.

An interesting idea due to Lipton [15], models the noisy channel as a computationally *bounded* adversarial channel. That is, the channel  $\mathcal{C}$  is modeled as a *probabilistic polynomial time* algorithm which can change symbols in the worst possible manner subject to an upper bound on the number of errors. Thus, Lipton's channels are essentially Hamming channels restricted to feasible computation. Modeling channels in this way makes a lot of sense as *all* real world channels are actually computationally bounded. The codes designed in this model guarantee that if a channel can cause incorrect decoding with high probability, it can also be used to break standard hardness assumptions.

<sup>1</sup> By good LDCs we mean LDCs with high information rate and high probability of error correction with small query size and constant error rate).

Working with such computationally bounded channels has led to several interesting results of late. In particular, Gopalan, Lipton, and Ding [9] develop a technique called *code scrambling* which recovers from high error rates by using few shared random bits. Similarly, Micali, Peikert, Sudan, and Wilson construct codes that can uniquely decode from error-rates beyond the classical bounds. Other notable results that use the idea of shared randomness in particular, are the results of Langberg [14] and Smith [21]. We remark that all these results are for standard error correcting codes and *not* for locally decodable codes. In this paper we continue in this important direction and construct good LDCs against computationally bounded noisy channels. We shall summarize our results shortly.

**PREVIOUS WORK ON LOCALLY DECODABLE CODES.** In order to understand the significance of our results, it is important that we shed some light on previous work related to locally decodable codes. Mainly, there has been two important research directions in this area: proving lower bounds on the size of LDCs and constructing good LDCs. All prior work in this area deals with computationally unbounded channels.

The first direction investigates the relation between the code length  $K$  and the message length  $k$  for  $(\ell, \rho, p)$ -LDCs. Katz and Trevisan [11] first started investigating this direction and showed that for non-adaptive LDCs,  $K$  is at least  $k^{1+\frac{1}{\ell-1}}$  (suppressing the dependence on  $\rho$  and  $p$ ). Deshpande et al [5] showed that this bound holds even for adaptive LDCs. Currently, the best known lower bounds for general locally decodable codes are due to Woodruff [24] who shows that  $K = \Omega\left(\frac{k^{1+\frac{2}{\ell-1}}}{\log k}\right)$ . A series of papers [8,17,19,12,23] concentrated on LDCs with  $\ell = 2$  (or 3) and established exponential lower bounds. In particular for 2-query LDCs  $K = \exp(\Omega(\frac{\rho}{2-2p}k))$ .

The other direction focussed on *constructing* the locally decodable codes. Important constructions in this direction for *constant* query length appeared in [2,3]. Unfortunately, all these constructions yield codes that are exponentially long in  $k_i$ . Currently, the best known construction is due to Yekhanin [25] who achieves locally decodable codes of sub-exponential length. For super-constant number of queries, however, better constructions are known. In particular, for  $\ell = (\log k)^{O(\frac{1}{p-0.5})}$  Babai et al [1] constructed LDCs of size  $K = k^{1+(p-0.5)}$ .

We derive following important conclusions from these results: all known constructions in the literature are either exponential in  $k$  or the query complexity is a huge polynomial in  $\log k$ . Furthermore, most of these constructions are able to provide only a *constant* probability of error correction which does not vanish with the size of the message.

**OUR RESULTS.** We consider the construction of locally decodable codes against computationally bounded channel. Under the *minimal* cryptographic assumption that one-way functions exist, we show how to construct *asymptotically good* locally decodable codes over a *binary* alphabet. Notice that small alphabet size is usually a requirement as most channels are best at transmitting only bits. Thus we have achieved locally decodable codes over binary alphabets with constant

information rate. This is already much better than all the known constructions in the literature. Our constructions require that the encoding and decoding algorithms share a secret key that is not known to the channel. For this reason we call our codes *private locally decodable codes*.

By reading at most  $\ell = \omega(\log^2 \kappa)$  bits in the codeword, our codes can correctly recover any given bit with probability  $p \geq 1 - \kappa^{-\omega(1)}$ , where  $\kappa$  is the security parameter, as long as the number of errors are less than a suitably chosen (constant) fraction. Thus, the probability of incorrect decoding is  $\kappa^{-\omega(1)}$  which is negligible in the security parameter.<sup>2</sup> Furthermore, if we allow the sender and the receiver to share a (synchronized) shared (such as a *public* counter), then our codes can have query complexity only  $\omega(\log \kappa)$ . We also show that  $\ell = \omega(\log \kappa)$  is necessary in order to achieve negligibly small probability of incorrect decoding. Thus, our codes have (almost) optimal query complexity.

Our codes are non-adaptive in nature. That is, the decoding procedure can make all its  $\ell$  queries at once without any dependence on the answers received from the corrupted word. This is a feature that might be desirable in some applications.

In some sense our results are incomparable to previous work because we work only against a computationally bounded adversary. But, a series of lower bound results and the poor information rate of best known constructions from previous work provide good motivation to study the problem in this new (weak yet reasonable) model.

ORGANIZATION. The rest of this article is organized as follows. The next section presents relevant background from coding theory and cryptography. We then describe our model which is followed by our constructions. Several details and discussions that have been omitted from this version because of space limitations can be found in the full version [18].

## 2 Definitions

In this section we will present relevant coding theory and cryptography. When dealing with codes, small alphabet size is usually preferred. Thus, unless specified otherwise, from now onwards we describe our constructions only for binary alphabets. It is straightforward to see their general version that has larger alphabet size. First we present some notations.

NOTATION. Vectors over  $\{0, 1\}$  will be represented in bold, e.g.,  $\mathbf{x}, \mathbf{y}$ . Because we are working over binary alphabets, occasionally we may refer to vectors over  $\{0, 1\}$  as (bit) strings. Concatenation of two vectors  $\mathbf{x}, \mathbf{y}$  is denoted by  $\mathbf{x} \circ \mathbf{y}$ . By  $[n]$  we denote the set of positive integers smaller than or equal to  $n$ :  $\{1, 2, \dots, n\}$ . A function  $\nu(n)$  is negligible in  $n$  if it vanishes faster than the inverse of every polynomial  $P(n)$  for a sufficiently large choice of  $n$ . Notation  $\Delta(\mathbf{x}, \mathbf{y})$  represents the hamming distance between vectors  $\mathbf{x}$  and  $\mathbf{y}$  which is the number of *alphabet*

<sup>2</sup> We can also choose the length of the input instead of the security parameter and then the error probability will be negligible in the length of the input.

positions in which they differ. By  $\mathbf{x}[j]$  we denote the  $j^{\text{th}}$  bit of  $\mathbf{x}$ . If  $S$  is a set then the process of selecting an element  $e$  from  $S$  uniformly at random, is denoted by:  $e \stackrel{\$}{\leftarrow} S$ . By  $\pi$  we denote a permutation (or a map) which permutes the bits of a given string  $\mathbf{x}$  by sending its  $j^{\text{th}}$  bit to the position  $\pi(j)$ . We will abuse the notation and denote by  $\pi(\mathbf{x})$  the string obtained by applying  $\pi$  to  $\mathbf{x}$  as above.

We now present some standard definitions, mostly taken from existing literature, e.g. [16].

**Definition 1 (Coding Scheme).** An  $(K, k)_q$ -coding scheme  $\mathcal{C} = (\mathcal{S}, \mathcal{R})$  over the alphabet  $\Sigma$  is a pair of encoding and decoding functions  $\mathcal{S} : \Sigma^k \rightarrow \Sigma^K$  and  $\mathcal{R} : \Sigma^K \rightarrow \Sigma^k$  for some positive integers  $K > k, q = |\Sigma| \geq 2$ . The (information) rate of the scheme, denoted  $R$ , is defined as  $R = \frac{k}{K}$ . The (minimum) distance of the coding scheme, denoted  $\delta$ , is defined as  $\delta = \min_{\mathbf{x}_1, \mathbf{x}_2 \in \Sigma^k} \Delta(\mathcal{S}(\mathbf{x}_1), \mathcal{S}(\mathbf{x}_2))$

In this paper we will be interested in asymptotic behavior of our codes. This requires us to consider infinite families of codes. Thus, we augment our current notation by indexing them and redefine the parameters such as the rate of the code.

**Definition 2 (Family of Coding Schemes).** Let  $\mathbb{C} = \{C_i\}_{i=1}^\infty$  be an infinite family of coding schemes where  $C_i$  is a  $(K_i, k_i)_{q_i}$ -coding scheme and  $\lim_{i \rightarrow \infty} K_i = \infty$ . The asymptotic information rate and minimum distance of  $\mathbb{C}$ , denoted  $R(\mathbb{C})$  and  $\delta(\mathbb{C})$  respectively, are defined as  $R(\mathbb{C}) = \liminf_{i \rightarrow \infty} k_i/K_i$  and  $\delta(\mathbb{C}) = \liminf_{i \rightarrow \infty} \delta_i/K_i$ . If  $\{\mathcal{S}_i\}$  and  $\{\mathcal{R}_i\}$  can be computed by two uniform probabilistic polynomial time algorithms, we say that the coding scheme is efficient.

In our constructions, as the encoding and decoding function do not change with  $i$  and only the parameters such as message length, code length etc. vary with  $i$ , we will drop the index  $i$  from  $\mathcal{S}, \mathcal{R}$ . Now we turn to the definition of standard locally decodable codes:

**Definition 3 (Locally Decodable Code).** An  $\ell$ -locally decodable code over a binary alphabet for error rate  $\rho$  and error-correction probability  $p > \frac{1}{2}$ , abbreviated as  $(\ell, \rho, p)$ -LDC, is a pair of probabilistic algorithms  $(\mathcal{S}, \mathcal{R})$ , where  $\mathcal{S} : \{0, 1\}^{k_i} \rightarrow \{0, 1\}^{K_i}$  and  $\mathcal{R}$  are the encoding and decoding algorithms respectively. If  $\mathbf{x} \in \{0, 1\}^{k_i}$  is the message and  $\mathbf{y} \leftarrow \mathcal{S}(\mathbf{x})$  is its encoding then we require that on input  $j \in [k_i]$ , the algorithm  $\mathcal{R}$  reads at most  $\ell$  bits from a given word  $\mathbf{y}'$  and outputs a bit  $b$  such that  $\Pr[b = \mathbf{x}[j]] \geq p$  provided that  $\Delta(\mathbf{y}, \mathbf{y}') \leq \rho K_i$  for some constant  $\rho$ .

Notice that locally decodable code is also a coding scheme as defined above but with the exception that the decoding algorithm does not have the whole codeword as input. It rather takes a single bit-position as input and is given oracular access to the codeword. Thus, terms such as a family of locally decodable coding schemes and asymptotic information rate are also defined analogously for locally decodable codes.



### 3 Our Model

We work in a shared key model where the encoding and decoding algorithms share some small secret information not known to the channel. In particular, this information will be the secret key to the pseudorandom permutation generator.

Deviating from traditional goals, we focus on constructing codes with high probability of recovering any given bit rather than some constant probability larger than  $1/2$ . In particular, we require the probability of *incorrect* decoding to be *negligible* in the message length. Of course small query complexity is desirable too along with negligible probability of incorrect decoding.

Because the encoding and decoding algorithms must share a key in our model, our codes are named private locally decodable codes. We present the definition of a private locally decodable code below.

**Definition 4 (Private  $\ell$ -Locally Decodable Code).** *Let  $\kappa$  be the security parameter. A private  $\ell$ -locally decodable code for a family of parameters  $\{(K_i, k_i)\}_{i=1}^{\infty}$  is a triplet of probabilistic polynomial time algorithms  $(\mathcal{K}, \mathcal{S}, \mathcal{R})$  such that:*

- $\mathcal{K}(1^\kappa)$  is the key generation algorithm that takes as input the security parameter  $\kappa$  and outputs a secret key  $sk$ .
- $\mathcal{S}(\mathbf{x}, sk)$  is the encoding algorithm that takes as input the message  $\mathbf{x}$  of length  $k_i = \text{poly}(\kappa)$  and the secret key  $sk$ . The algorithm outputs  $\mathbf{y} \in \{0, 1\}^{K_i}$  that denotes an encoding of  $\mathbf{x}$ .
- $\mathcal{R}(j, sk)$  denotes the decoding algorithm, which takes as input a bit position  $j \in [k_i]$  and the secret key  $sk$ . It outputs a single bit  $b$  denoting the decoding of  $\mathbf{x}[j]$  by making at most  $\ell$  (adaptive) queries into a given a codeword  $\mathbf{y}'$  possibly different from  $\mathbf{y}$ .

The information rate of the scheme is  $\liminf_{i \rightarrow \infty} k_i / K_i$ .

Parameter  $\ell$  is called the query complexity of the code. Notice that in our definition, the decoding algorithm is supposed to have the same secret key  $sk$  as was used to encode the message. Obviously this definition does not make sense until we introduce the probability of correctly obtaining  $\mathbf{x}[j]$  using the decoding procedure. But before that, we need to explain the game between the channel and the encoding and decoding algorithms.

A computationally bounded adversarial channel  $\mathcal{C}$  with error rate  $\rho$  is a probabilistic polynomial time algorithm which repeatedly interacts with the encoding algorithm  $\mathcal{S}$  and the decoding algorithm  $\mathcal{R}$  polynomially many times until it terminates. Each iteration takes place as follows:

1. Given a security parameter  $\kappa$ , the key generation algorithm outputs a secret key  $sk \leftarrow \mathcal{K}(1^\kappa)$ . The secret is given to both  $\mathcal{S}, \mathcal{R}$  but not to the channel. The channel is given  $\kappa$ .
2. In  $h^{\text{th}}$  iteration, the channel  $\mathcal{C}$  chooses a message  $\mathbf{x}^{(h)} \in \{0, 1\}^{k_i}$  and hands it to the sender.
3. The sender computes  $\mathbf{y}^{(h)} \leftarrow \mathcal{S}(\mathbf{x}^{(h)}, sk)$  and hands the codeword  $\mathbf{y}^{(h)} \in \{0, 1\}^{K_i}$  back to the channel.

4. The channel corrupts at most a fraction  $\rho$  of all  $K_i$  bits in  $\mathbf{y}^{(h)}$  to output the corrupted codeword  $\mathbf{y}'^{(h)}$ , i.e.,  $\Delta(\mathbf{y}^{(h)}, \mathbf{y}'^{(h)}) \leq \rho K_i$ . It gives  $\mathbf{y}'^{(h)}$  and a challenge bit  $j$  to the receiver  $\mathcal{R}$ .
5. The receiver makes at most  $\ell$  (possibly adaptive) queries into the new codeword  $\mathbf{y}'^{(h)}$  and outputs  $b \leftarrow \mathcal{R}(j, sk)$ .

We say that a code  $(\mathcal{K}, \mathcal{S}, \mathcal{R})$  *correctly decodes from error rate  $\rho$  with high probability* if for all probabilistic polynomial time algorithms  $\mathcal{C}$  in the above experiment, for all messages  $\mathbf{x} \in \{0, 1\}^{k_i}$ , and for all  $j \in [k_i]$  we have that  $\Pr[b \neq \mathbf{x}^{(h)}[j]] = \nu(\kappa)$ , where the probability is taken over the random coins of  $\mathcal{K}, \mathcal{S}, \mathcal{R}$ , and  $\mathcal{C}$ .

In above definition, we have that maximum value of  $h$  is bounded from above by a value polynomial in the length of the input. If we have a code that only works (i.e., correctly decodes from error rate  $\rho$  with high probability) *once* (i.e., only for  $h = 1$ ) and guarantees nothing for repeated executions, we call such a private locally decode to be *one time*.

In the above definition, we assume that the adversary always sends messages of the same length known a priori both to the sender and receiver. We stress that it is merely a *technicality*. If one wants that the adversary be given the flexibility to choose the message lengths, then also our constructions work but with a slight technical modification<sup>3</sup>.

## 4 Our Constructions

In this section we provide our constructions. We do this in two stages. First we provide two constructions which work only once, i.e., they are one-time. First such construction is a simple repetition code with  $\log^2 \kappa$  query complexity<sup>4</sup> and the second one is based on any asymptotically good code and has the same query complexity but a better (i.e., asymptotically positive) information rate. In the second stage, we show how to uplift our construction so that we get a code that works for polynomially many invocations, i.e., satisfies our actual definition.

Although we describe our construction for  $\log^2 \kappa$  query complexity, they actually work for any query complexity that grows faster than  $\log \kappa$ , (i.e.,  $\omega(\log \kappa)$ ). We also show that  $\omega(\log \kappa)$  query complexity is essential if we want decoding error to be negligible in  $\kappa$ . Thus our constructions have optimal query length.

### 4.1 Constructions for One-Time Codes

A SIMPLE REPETITION CODE. Our first code is a very simple repetition code. Let  $\mathbf{x}$  be the string we want to encode. Our repetition code  $(\mathcal{K}^{\text{REP}}, \mathcal{S}^{\text{REP}}, \mathcal{R}^{\text{REP}})$  is as follows.

<sup>3</sup> See full version [18].

<sup>4</sup> Technically, the query complexity is actually  $\lceil \log^2 \kappa \rceil$ , but in order to avoid the cluttering in presentation, we shall drop floors and ceiling in formulas. This does not affect our analysis.

**Algorithm**  $\mathcal{K}^{\text{REP}}(1^\kappa)$ . This is a randomized algorithm which simply outputs a truly random permutation  $\pi$  and a truly random mask  $\mathbf{r}$  both of size  $K_i$  (the code length, to be stated later) Thus,  $sk \leftarrow (\pi, \mathbf{r})$ .

**Algorithm**  $\mathcal{S}^{\text{REP}}(\mathbf{x}, sk)$ . The algorithm works as follows:

- Compute  $\mathbf{x}'$  by repeating each bit of  $\mathbf{x}$  for  $\log^2 \kappa$  times.
- Compute  $\mathbf{y}_1 \leftarrow \pi(\mathbf{x}')$  and output  $\mathbf{y} = \mathbf{y}_1 \oplus \mathbf{r}$ .

Notice that the size of codeword  $y$  is  $K_i = k_i \log^2 \kappa$ .

**Algorithm**  $\mathcal{R}^{\text{REP}}(j, sk)$ . To decode, the algorithm simply reads all  $\ell = \log^2 \kappa$  bit positions of corrupted word  $\mathbf{y}'$  that correspond to bit position  $j$  of the original message  $\mathbf{x}$ , and decides by majority after unmasking them with  $\mathbf{r}$ . Note that computing these bit positions requires reading only  $\ell$  entries from the stored permutation  $\pi$  and hence has *polylogarithmic* running time. The algorithm works as follows:

- Let  $j_1, j_2, \dots, j_\ell$  denote the  $\ell$  bit positions of  $\mathbf{x}'$  that have the copies of  $\mathbf{x}[j]$ . Compute  $i_h \leftarrow \pi(j_h)$  for  $h = 1, 2, \dots, \ell$ .
- Compute  $\mathbf{y}'[i_1] \oplus \mathbf{r}[i_1], \mathbf{y}'[i_2] \oplus \mathbf{r}[i_2], \dots, \mathbf{y}'[i_\ell] \oplus \mathbf{r}[i_\ell]$  and output the majority bit.

Notice that the query complexity is  $\ell = \log^2 \kappa$ .

In the above, instead of  $\ell = \log^2 \kappa$  we can choose any  $\ell = \omega(\log \kappa)$ .

**Theorem 1.** *There exists a constant  $\rho$  such that  $(\mathcal{K}^{\text{REP}}, \mathcal{S}^{\text{REP}}, \mathcal{R}^{\text{REP}})$  is a one-time private  $\omega(\log \kappa)$ -locally decodable code that correctly decodes from error rate  $\rho$  with high probability.*

PROOF. It is easy to see that a bit is decoded incorrectly if and only if at least  $\lambda = \ell/2$  of its  $\ell$  copies were corrupted. From Lipton’s theorem [15], it follows that if the permutation  $\pi$  and the mask  $\mathbf{r}$  are truly random, then the adversarial channel  $\mathcal{C}$  behaves like a binary symmetric channel which corrupts at most a fraction  $\rho$  of all bits. Thus, the probability  $p$  of incorrect decoding for a given bit position  $j$  can be calculated by a simple combinatorial analysis:

$$p < \frac{\binom{\ell}{\lambda} \binom{n-\lambda}{m-\lambda}}{\binom{n}{m}} < \left( 256 \frac{\ell}{\lambda} \cdot e^{b+1} \rho \right)^\lambda \quad (\text{see full version})$$

which is less than  $2^{-\ell} = \nu(k)$  for  $\rho = \frac{1}{2 \cdot 11 \cdot e^{b+1}}$  and  $\ell = \omega(\log \kappa)$ . Because probability of incorrectly decoding a given bit is negligible and there are only  $k_i = \text{poly}(\kappa)$  bits, we conclude that probability of incorrectly decoding *any* bit is negligible given that the permutation  $\pi$  and  $\mathbf{r}$  are truly random (which is the case). ■

CONSTRUCTION BASED ON ANY ASYMPTOTICALLY GOOD CODE. In this section we present the construction of a locally decodable code based on any asymptotically good code. We will present a general construction and its analysis without setting the parameters explicitly. Later on, we will set the parameters suitably so as to obtain a locally decodable code satisfying our goals. We start with the definition of asymptotically good codes.

**Definition 5 (Asymptotically Good Codes).** *A family of codes  $\mathbb{C} = \{C_i\}_{i=1}^\infty$  is said to be asymptotically good if  $R(\mathbb{C}), \delta(\mathbb{C}) > 0$ .*

We remark that efficient asymptotically good codes are known [10,20]. Sometimes we may simply use  $R$  and  $\delta$  and drop the argument  $\mathbb{C}$  when it is clear from the context. Also, from now on, in our constructions we will only refer to  $C = (\mathcal{S}, \mathcal{R})$  which is a  $(A, a)_q$  coding scheme from the family of asymptotically good codes. Let  $\frac{1}{\beta}$  be the rate of the code so that  $A \leq \beta a$ , where  $\beta$  is a constant. Let  $\gamma$  denote the constant fraction such that  $C$  can recover from error rate  $\gamma$  (i.e. the number of errors allowed is equal to  $\gamma A$  symbols). Because we are working over an alphabet of size  $q$ , let  $c = \log q$ , and we will sometimes say that the message  $\mathbf{x}$  is a sequence of  $c \cdot a$  bits and the codeword  $\mathbf{y}$  is a sequence of  $c \cdot A$  bits. A symbol is considered corrupted if any of its  $c$  bits gets corrupted and hence number of bit-errors  $e$  from which  $C$  can recover is still at most  $\gamma A$ .

OUR CONSTRUCTION. On a high level, we visualize the message  $\mathbf{x}$  as a series of  $n_i$  messages each of which will contain  $a$  symbols from  $\Sigma$ , or in other words each message will contain  $a$  blocks of  $c = \log q$  bits each. That is,

$$\mathbf{x} = \overbrace{(\mathbf{x}_1 \circ \dots \circ \mathbf{x}_a)}^1 \circ \overbrace{(\mathbf{x}_{a+1} \circ \dots \circ \mathbf{x}_{2a})}^2 \circ \dots \circ \overbrace{(\mathbf{x}_{(n_i-1)a+1} \circ \dots \circ \mathbf{x}_{n_i a})}^{n_i}$$

Now each message (contained in parentheses) will be encoded using the encoding function  $\mathcal{S}$  of the asymptotically good coding scheme  $C$  and all such encodings will be concatenated together. The resulting string will be permuted according to a pseudo-random permutation  $\pi$  and XORed with a pseudorandom string  $\mathbf{r}$  to yield the final code. Notice that the message length for our locally decodable code is:  $k_i = |\mathbf{x}| = c \cdot a \cdot n_i$ . We will choose the parameters  $A, a$  for the asymptotically good code  $C$  in such a way that we will achieve locally decodable codes with desirable properties. Following is the formal description of our code.

Let  $C$  be the asymptotically good (aG) code with  $a = \log^2 \kappa$  where  $\kappa$  is the security parameter. Let the rate of the code be  $1/\beta$  and error-tolerance  $\gamma$  so that code length  $A = \beta a$  and it can correct up to  $\gamma A$  errors. Following is the set of algorithms.

**Algorithm**  $\mathcal{K}^{\text{aG}}(1^\kappa)$ . Same as for the repetition code:  $sk \leftarrow (\pi, \mathbf{r})$

**Algorithm**  $\mathcal{S}^{\text{aG}}(\mathbf{x}, sk)$ . The algorithm works as follows:

- Let  $\mathbf{x} = \mathbf{w}_1 \circ \mathbf{w}_2 \circ \dots \circ \mathbf{w}_{n_i}$ , where  $\mathbf{w}_s = \mathbf{x}_{(s-1)a+1} \circ \mathbf{x}_{(s-1)a+2} \circ \dots \circ \mathbf{x}_{sa}$  for  $s = 1, 2, \dots, n_i$ . Notice that  $k_i = ca \cdot n_i$ .
- Each  $\mathbf{w}_s$  is a sequence of  $a$  symbols from  $\Sigma$ . Encode each  $\mathbf{w}_s$  using the encoding function  $\mathcal{S}$  of  $C$  to get encoded words  $\mathbf{w}'_s$ . That is, for each  $s$ , compute:  $\mathbf{w}'_s \leftarrow \mathcal{S}(\mathbf{w}_s)$
- Let  $\mathbf{x}' = \mathbf{w}'_1 \circ \mathbf{w}'_2 \circ \dots \circ \mathbf{w}'_{n_i}$ . Compute  $\mathbf{y}_1 \leftarrow \pi(\mathbf{x}')$  and output  $\mathbf{y}_1 \oplus \mathbf{r}$ .

Notice that the size of codeword  $y$  is  $K_i = cAn_i = \frac{A}{a}k_i = \beta k_i$ .

**Algorithm**  $\mathcal{R}^{\text{aG}}(j, sk)$ . The  $j^{\text{th}}$  bit of message  $\mathbf{x}$  lies in  $\mathbf{w}_s$  where  $s = \lceil \frac{j}{ac} \rceil$ . The decoding algorithm simply reads all the  $cA$  bits (corresponding to  $\mathbf{w}'_s$ ) from the (possibly) corrupted encoding  $\mathbf{y}'$  using  $\ell = cA$  queries, un.masks them using  $\mathbf{r}$  and then decodes using the decoding algorithm  $\mathcal{R}$  to obtain the complete subsequence  $\mathbf{w}_s$ . Notice that positions of all  $cA$  bits corresponding to  $\mathbf{w}'_s$  can be computed using  $\pi$  in sublinear time.

- Let  $j_1, j_2, \dots, j_\ell$  be the bit positions corresponding to the bits of  $\mathbf{w}'_s$ . Then for all  $h = 1, 2, \dots, \ell$  compute  $i_h \leftarrow \pi(j_h)$ .
- Compute  $\mathbf{y}'[i_1] \oplus \mathbf{r}[i_1], \mathbf{y}'[i_2] \oplus \mathbf{r}[i_2], \dots, \mathbf{y}'[i_\ell] \oplus \mathbf{r}[i_\ell]$  and obtain points  $\mathbf{w}'_s$  (possibly corrupted).
- Apply the decoding algorithm  $\mathcal{R}$  instance on possibly corrupted  $\mathbf{w}'_s$  to obtain  $\mathbf{w}_s$ . Output that bit of  $\mathbf{w}_s$  which corresponds to the  $j^{\text{th}}$  bit of  $\mathbf{x}$ .

Notice that the query complexity is  $\ell = cA$ .

Above code is a private locally decodable code with *positive* information rate  $\liminf_{i \rightarrow \infty} \frac{k_i}{K_i} = \frac{1}{\beta}$  and query complexity  $\ell = cA = \log q \cdot \beta \log^2 \kappa = O(\log^2 \kappa)$ . Notice that we can use  $a = \omega(\log \kappa)$  and achieve  $\ell = \omega(\log \kappa)$ . Let us now prove the following.

**Theorem 2.** *There exists a constant  $\rho$  such that  $(\mathcal{K}^{\text{aG}}, \mathcal{S}^{\text{aG}}, \mathcal{R}^{\text{aG}})$  is a one-time private  $\omega(\log \kappa)$ -locally decodable code with constant information rate that correctly decodes from error rate  $\rho$  with high probability.*

PROOF. We have already proved the claims about information rate and query complexity. We only need to show that the code indeed correctly recovers from some constant error rate  $\rho$  with high probability.

Notice that the algorithm may decode a given bit  $j$  incorrectly only if  $\mathbf{w}'_s$  is corrupted in at least  $\lambda = \gamma A$  bit positions. This is because  $C$  can correctly recover from error rate  $\gamma$ . We thus need to bound the probability that more than  $\lambda$  bits of  $\mathbf{w}'_s$  are flipped by any adversary. As  $\pi$  and  $\mathbf{r}$  are truly random we can use Lipton’s theorem, and bound this probability just by analyzing the code represented by  $\mathbf{x}'$  in the presence of a binary symmetric channel<sup>5</sup> which only corrupts at most a  $\rho$  fraction of all  $K_i$  bits. Now the probability  $p$  of incorrectly decoding bit  $j$  can be bounded as before for the repetition code and comes out to be (see full version)  $\nu(\kappa)$ . ■

## 4.2 Final Construction

In this section, we now show how to uplift our one-time constructions so that they work for polynomially many times. The idea is to divide the codeword obtained from one-time code into small chunks and then encrypt each chunk using a suitable encryption scheme. This way, we hide the permutation  $\pi$  behind the encryption scheme and hence can use just that permutation every time we encode. Notice that encryption can blow up the size of the chunk by a factor of the security parameter  $\kappa$ . Thus, we instead encrypt the chunk by a pseudorandom one-time pad obtained from a pseudorandom function<sup>7</sup>. In order to tolerate the errors in these ciphertexts, we encode each ciphertext using an off the shelf asymptotically good error-correcting code. Details follow.

Let  $f_{\text{KEY}}$  denote a pseudorandom function with the key KEY. Let  $(\mathcal{K}^{\text{aG}}, \mathcal{S}^{\text{aG}}, \mathcal{R}^{\text{aG}})$  be the one-time coding scheme that we will use as our base. Let  $\log^2 \kappa$  where  $\kappa$  is the security parameter<sup>6</sup>. Our final private locally decodable code  $(\mathcal{K}^{\text{FIN}}, \mathcal{S}^{\text{FIN}}, \mathcal{R}^{\text{FIN}})$  is as follows:

<sup>5</sup> Recall that BSC introduces errors randomly with some fixed error probability.

<sup>6</sup> Any  $a = \omega(\log \kappa)$  would also work for our constructions.

**Algorithm**  $\mathcal{K}^{\text{FIN}}(1^\kappa)$ . This algorithm first runs  $\mathcal{K}^{\text{aG}}((1^\kappa))$  to obtain  $sk'$  and then chooses a truly random seed KEY of size  $\log^2 \kappa$  for the pseudorandom function  $f$ . It sets  $sk \leftarrow (sk', \text{KEY})$ .

**Algorithm**  $\mathcal{S}^{\text{FIN}}(\mathbf{x}, sk)$ . The algorithm works as follows:

- Obtain  $y' \leftarrow \mathcal{S}^{\text{aG}}(\mathbf{x}, sk')$ . Let  $K'_i = |y'|$ . Divide  $y'$  into chunks  $B_1, B_2, \dots, B_z$  of size  $a$  each where  $z = K'_i/a$ . Now, *encrypt* each  $B_h$  as  $E_h = (r_h, f_{\text{KEY}}(r_h) \oplus B_h)$  where  $h \in [z]$  and  $r_h$  is a string of length  $a$  chosen uniformly at random.
- Now, *encode* each ciphertext  $E_h$  using an *asymptotically good* code of information rate  $1/\beta_1$ :  $F_h \leftarrow \mathcal{S}(E_h)$ . Let  $y = F_1 \circ F_2 \circ \dots \circ F_z$ . Notice that  $|y| = 2\beta_1 K'_i$ . Output  $y$ .

Notice that the size of codeword  $y$  is  $K_i = 2\beta_1 K'_i = 2\beta\beta_1 k_i$ .

**Algorithm**  $\mathcal{R}^{\text{FIN}}(j, sk)$ . The decoding algorithm will first run the  $\mathcal{R}^{\text{aG}}(j, sk)$  and let  $j_1, j_2, \dots, j_{\ell'}$  denote the indexes queried by  $\mathcal{R}^{\text{aG}}$  (where  $\ell'$  is query length of the one-time code). From the construction, these queries are actually queries into the intermediate code  $y'$ . Let  $B_{j_h}$ ,  $0 \leq h \leq \ell$ , denote that chunk of  $y'$  in which the  $j_h^{\text{th}}$  bit of  $y'$  lies. Then,  $\mathcal{R}^{\text{FIN}}$  reads all bits of  $y'$  corresponding to each block  $F_{j_h}$ , for  $j_1, j_2, \dots, j_{\ell'}$ . Thus the query length is  $\ell = a\ell'$ . Note that these blocks may be corrupted. Now algorithm proceeds as follows:

- Decode each block  $F_{j_h}$  using the decoding algorithm  $\mathcal{R}$  to obtain (possibly incorrect) blocks  $E_{j_h} = (r_{j_h}, E'_{j_h})$ . Now compute  $B_{j_h} = E'_{j_h} \oplus f_{\text{KEY}}(r_{j_h})$ . Notice that  $B_{j_h}$  may be totally different from what it was originally when encoded. Read that bit of  $B_{j_h}$  that corresponds to  $j_h^{\text{th}}$  bit of  $y'$  and give it to  $\mathcal{R}^{\text{aG}}$  when asked.
- Return whatever is returned by  $\mathcal{R}^{\text{aG}}$ .

Notice that the query complexity is  $\ell = a\ell'$ .

Above code is a private locally decodable code with *positive* information rate  $\frac{1}{2\beta\beta_1}$  and query complexity  $\ell = a\ell'$ . As,  $\ell' = \omega(\log \kappa)$  we could have used any  $a = \omega(\log \kappa)$ , we have a code with query complexity  $\omega(\log^2 \kappa)$ . We have (see full version for proofs & discussions):

**Theorem 3.** *There exists a constant  $\rho$  such that  $(\mathcal{K}^{\text{FIN}}, \mathcal{S}^{\text{FIN}}, \mathcal{R}^{\text{FIN}})$  is a private  $\omega(\log^2 \kappa)$ -locally decodable code with constant information rate that correctly decodes from error rate  $\rho$  with high probability.*

**Lemma 1.** *Private locally decodable codes with query complexity  $O(\log \kappa_i)$  (or smaller) that decode from constant error rate with high probability do not exist.*

## References

1. Babai, L., Fortnow, L., Levin, L.A., Szegedy, M.: Checking computations in poly-logarithmic time. In: STOC, pp. 21–31 (1991)
2. Beimel, A., Ishai, Y.: Information-theoretic private information retrieval: A unified construction. In: ICALP, pp. 912–926 (2001)

3. Beimel, A., Ishai, Y., Kushilevitz, E., Raymond, J.-F.: Breaking the  $o(n^{1/(2k-1)})$  barrier for information-theoretic private information retrieval. In: FOCS, pp. 261–270 (2002)
4. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. *J. ACM* 45(6), 965–981 (1998)
5. Deshpande, A., Jain, R., Kavitha, T., Radhakrishnan, J., Lokam, S.V.: Better lower bounds for locally decodable codes. In: IEEE Conference on Computational Complexity, pp. 184–193. IEEE Computer Society Press, Los Alamitos (2002)
6. Gemmell, P., Lipton, R.J., Rubinfeld, R., Sudan, M., Wigderson, A.: Self-testing/correcting for polynomials and for approximate functions. In: STOC, pp. 32–42 (1991)
7. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. *J. ACM* 33(4), 792–807 (1986)
8. Goldreich, O., Karloff, H.J., Schulman, L.J., Trevisan, L.: Lower bounds for linear locally decodable codes and private information retrieval. In: IEEE Conference on Computational Complexity, pp. 175–183. IEEE Computer Society Press, Los Alamitos (2002)
9. Gopalana, P., Lipton, R.J., Ding, Y.Z.: Error correction against computationally bounded adversaries. In: Manuscript (2004)
10. Justesen, J.: A class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory* 18, 652–656 (1972)
11. Katz, J., Trevisan, L.: On the efficiency of local decoding procedures for error-correcting codes. In: STOC, pp. 80–86 (2000)
12. Kerenidis, I., de Wolf, R.: Exponential lower bound for 2-query locally decodable codes via a quantum argument. In: STOC, pp. 106–115 (2003)
13. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS, pp. 364–373 (1997)
14. Langberg, M.: Private codes or succinct random codes that are (almost) perfect. In: FOCS, pp. 325–334 (2004)
15. Lipton, R.J.: A new approach to information theory. In: STACS, pp. 699–708 (1994)
16. Micali, S., Peikert, C., Sudan, M., Wilson, D.A.: Optimal Error Correction Against Computationally Bounded Noise. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, Springer, Heidelberg (2006)
17. Obata, K.: Optimal lower bounds for 2-query locally decodable linear codes. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 39–50. Springer, Heidelberg (2002)
18. Ostrovsky, R., Pandey, O., Sahai, A.: Private Locally Decodable Codes. Available at <http://eprint.iacr.org/2007/025/>
19. Shiohata, D., Lokam, S.V.: An optimal lower bound for 2-query locally decodable linear codes. *Inf. Process. Lett.* 97(6), 244–250 (2006)
20. Sipser, M., Spielman, D.A.: Expander codes. In: FOCS, pp. 566–576 (1994)
21. Smith, A.: Scrambling adversarial errors using few random bits. In: SODA (2007)
22. Sudan, M.: Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems, PhD Thesis, University of California at Berkeley (1992)
23. Wehner, S., de Wolf, R.: Improved lower bounds for locally decodable codes and private information retrieval. In: ICALP, pp. 1424–1436 (2005)
24. Woodruff, D.: New lower bounds for general locally decodable codes. In: ECCV TR07-006 (2007)
25. Yekhanin, S.: Towards 3-query locally decodable codes of subexponential length. In: STOC (2007) Also appears on ECCV as TR06-127 under a different title.

# Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms

Mihir Bellare and Thomas Ristenpart

Dept. of Computer Science & Engineering 0404, University of California San Diego  
9500 Gilman Drive, La Jolla, CA 92093-0404, USA  
{mihir, tristenp}@cs.ucsd.edu  
<http://www-cse.ucsd.edu/users/{mihir, tristenp}>

**Abstract.** In the dedicated-key setting, one uses a compression function  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  to build a family of hash functions  $H^f: \mathcal{K} \times \mathcal{M} \rightarrow \{0, 1\}^n$  indexed by a key space  $\mathcal{K}$ . This is different from the more traditional design approach used to build hash functions such as MD5 or SHA-1, in which compression functions and hash functions do not have dedicated key inputs. We explore the benefits and drawbacks of building hash functions in the dedicated-key setting (as compared to the more traditional approach), highlighting several unique features of the former. Should one choose to build hash functions in the dedicated-key setting, we suggest utilizing multi-property-preserving (MPP) domain extension transforms. We analyze seven existing dedicated-key transforms with regard to the MPP goal and propose two simple new MPP transforms.

## 1 Introduction

**TWO SETTINGS.** A popular method for designing hash functions proceeds as follows. First, one designs a compression function  $f: \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$ , where  $d$  is the length of a data block and  $n$  is the length of the chaining variable. Then one specifies a *domain extension transform*  $H$  that utilizes  $f$  as a black box to implement the hash function  $H^f: \mathcal{M} \rightarrow \{0, 1\}^n$  associated to  $f$ , where  $\mathcal{M}$  is some large message space. Most in-use hash functions, for example the MD-x family [21] and SHA-1 [19], were constructed using this approach.

There also exists a second setting for hash function design and analysis, in which compression functions and hash functions both have a *dedicated key input*. A dedicated-key compression function has signature  $f: \{0, 1\}^k \times \{0, 1\}^{d+n} \rightarrow \{0, 1\}^n$ . A transform  $H$  now uses  $f(\cdot, \cdot)$  as a black-box to implement a family of hash functions  $H^f: \mathcal{K} \times \mathcal{M} \rightarrow \{0, 1\}^n$  indexed by a key space  $\mathcal{K}$ . We call this the *dedicated-key setting*. Note that although we use the term “key”, this does not mean that a key  $K \in \mathcal{K}$  is necessarily private. Indeed, hash functions often *need* to be publicly computable (e.g., for verifying digital signatures) and so in these settings every party must have access to the key.

**THIS PAPER.** Due to recent collision-finding attacks against in-use hash functions such as MD5 and SHA-1 [26,27], new hash functions are going to be designed



and standardized. A crucial choice for designers will be whether one should build hash functions in the first setting (continuing in the current tradition of in-use hash functions) or in the dedicated-key setting. Our first contribution is to present relative merits of the two settings described above, pointing out several important benefits of the dedicated-key setting, but also its most significant drawbacks.

Should one choose to work in the dedicated-key setting, the natural next question is how to best build hash functions in it. Because hash functions are currently used in a wide variety of applications with disjoint security requirements, we suggest building hash functions using *multi-property-preserving* (MPP) transforms, introduced for the non-dedicated-key setting in [6]. An MPP transform  $H$  simultaneously *preserves* numerous properties of interest: if the compression function  $f$  has security property  $P$ , then  $H^f$  has  $P$  also. Our second contribution is an MPP-orientated analysis of several dedicated-key transforms and the proposal of two new transforms that better meet the MPP goal.

We now briefly summarize our results in more detail.

**THE DEDICATED-KEY SETTING.** In Section 3, we discuss the dedicated-key setting, pointing out several features which are distinct from the more traditional setting. We describe two important benefits of the dedicated-key setting: hash function heterogeneity (allowing users to specify independent instances of the hash function) and improved security guarantees (particularly for message authentication, a wide-spread application of hash functions). On the other hand, a significant downside of dedicated keys is a decrease in efficiency.

**DEDICATED-KEY TRANSFORMS.** In Section 5, we provide an MPP-orientated treatment of transforms in the dedicated-key setting, analyzing seven previously proposed Merkle-Damgård-like transforms: plain Merkle-Damgård (MD) [16,12], strengthened MD (sMD) [12], prefix-free MD (Pre) [15], Shoup’s transform (Sh) [24], the strengthened Nested Iteration transform (sNI) [1], the Nested Iteration transform (NI) [15], and the Chain-Shift transform (CS) [15]. Figure 1 summarizes our results for the existing seven transforms. For each transform we determine if it is collision-resistance preserving (CR-Pr), message authentication code preserving (MAC-Pr), pseudorandom function preserving (PRF-Pr), and pseudorandom oracle preserving (PRO-Pr). A “Yes” in the P-Pr column for transform  $T$  means that, if a compression function  $f$  has property  $P$ , then  $T^f$  provably has property  $P$ . A “No” means that there exists a compression function  $f$  with property  $P$ , but for which  $T^f$  does *not* have  $P$ . Only one of the seven transforms preserves the first four properties (though requiring two keys to do so), and so we suggest a new MPP transform, called Strengthened Chain-Shift, which is efficient and requires just one key.

We also investigate the property of being a universal one-way hash function [18], which we’ll call target-collision resistance (following [10]). The practical value of TCR-Pr transforms is limited by the significant amount of key material they require; see Section 5.5 for a discussion. That said, none of the transforms thus far preserve it along with the other four properties, and so we suggest a new transform, Enveloped Shoup, which preserves all five properties.

|     | CR-Pr                       | MAC-Pr                   | PRF-Pr | PRO-Pr                   | TCR-Pr                   | Efficiency $\tau(L)$        | Key bits                             |
|-----|-----------------------------|--------------------------|--------|--------------------------|--------------------------|-----------------------------|--------------------------------------|
| MD  | No <a href="#">[16,12]</a>  | No                       | Yes    | No <a href="#">[11]</a>  | No <a href="#">[10]</a>  | $\lceil (L+1)/d \rceil$     | $k$                                  |
| sMD | Yes <a href="#">[16,12]</a> | No                       | Yes    | No <a href="#">[11]</a>  | No <a href="#">[10]</a>  | $\lceil (L+65)/d \rceil$    | $k$                                  |
| Pre | No                          | Yes <a href="#">[15]</a> | Yes    | Yes <a href="#">[11]</a> | No                       | $\lceil (L+1)/(d-1) \rceil$ | $k$                                  |
| Sh  | Yes <a href="#">[24]</a>    | No                       | Yes    | No                       | Yes <a href="#">[24]</a> | $\lceil (L+65)/d \rceil$    | $k + n \lceil \log_2 \tau(L) \rceil$ |
| sNI | Yes                         | Yes <a href="#">[1]</a>  | Yes    | Yes <a href="#">[6]</a>  | No                       | $\lceil (L+65)/d \rceil$    | $2k$                                 |
| NI  | No                          | Yes <a href="#">[15]</a> | Yes    | Yes <a href="#">[6]</a>  | No                       | $\lceil (L+1)/d \rceil$     | $2k$                                 |
| CS  | No                          | Yes <a href="#">[15]</a> | Yes    | Yes <a href="#">[6]</a>  | No                       | $\lceil (L+1+n)/d \rceil$   | $k$                                  |
| sCS | Yes                         | Yes <a href="#">[15]</a> | Yes    | Yes <a href="#">[6]</a>  | No                       | $\lceil (L+65+n)/d \rceil$  | $k$                                  |
| ESh | Yes                         | Yes                      | Yes    | Yes                      | Yes                      | $\lceil (L+65+n)/d \rceil$  | $k + n \lceil \log_2 \tau(L) \rceil$ |

**Fig. 1.** Summary of transforms in the dedicated-key setting when applied to a compression function  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$ . Bold-faced claims are novel. Efficiency is measured by  $\tau(L)$ , the number of compression function applications used to hash an  $L$ -bit string.

## 2 Notation and Definitions

NOTATION. We denote pairwise concatenation by  $\parallel$ , e.g.  $M \parallel M'$ , and write  $M_1 \cdots M_k$  to mean  $M_1 \parallel M_2 \parallel \dots \parallel M_k$ . For brevity, we define the following semantics for the notation  $M_1 \cdots M_k \stackrel{d}{\leftarrow} M$  where  $M$  is a string of bits: 1) define  $k = \lceil |M|/d \rceil$  and 2) if  $|M| \bmod d = 0$  then parse  $M$  into  $M_1, M_2, \dots, M_k$  where  $|M_i| = d$  for  $1 \leq i \leq k$ , otherwise parse  $M$  into  $M_1, M_2, \dots, M_{k-1}, M_k$  where  $|M_i| = d$  for  $1 \leq i \leq k-1$  and  $|M_k| = |M| \bmod d$ . For any finite set  $S$  we write  $s \stackrel{\$}{\leftarrow} S$  to signify uniformly choosing a value  $s \in S$ . A random oracle is an algorithm  $\text{RF}_{Dom, Rng}$  that, on input  $X \in Dom$ , returns a value  $Y \stackrel{\$}{\leftarrow} Rng$ . Repeat queries are, however, answered consistently. We sometimes write  $\text{RF}_{d,r}$  when  $Dom = \{0, 1\}^d$  and  $Rng = \{0, 1\}^r$ .

SECURITY NOTIONS. Let  $F: \mathcal{K} \times Dom \rightarrow Rng$  be a function with non-empty key space  $\mathcal{K}$  and define  $F_K(\cdot) = F(K, \cdot)$ . Then we define the following security experiments:

- tcr:  $\epsilon = \Pr \left[ (X, S) \stackrel{\$}{\leftarrow} \mathcal{A}_1, K \stackrel{\$}{\leftarrow} \mathcal{K}, X' \stackrel{\$}{\leftarrow} \mathcal{A}_2(S, K) : \begin{matrix} X \neq X' \wedge \\ F_K(X) = F_K(X') \end{matrix} \right]$
- cr:  $\epsilon = \Pr \left[ K \stackrel{\$}{\leftarrow} \mathcal{K}, (X, X') \stackrel{\$}{\leftarrow} \mathcal{A}(K) : X \neq X' \wedge F_K(X) = F_K(X') \right]$
- mac:  $\epsilon = \Pr \left[ K \stackrel{\$}{\leftarrow} \mathcal{K}, (X, T) \stackrel{\$}{\leftarrow} \mathcal{A}^{F(K, \cdot)} : F_K(X) = T \wedge X \text{ not queried} \right]$
- prf:  $\epsilon = \Pr \left[ K \stackrel{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{F(K, \cdot)} \Rightarrow 1 \right] - \Pr \left[ \rho \stackrel{\$}{\leftarrow} \text{Func}(Dom, Rng) : \mathcal{A}^{\rho(\cdot)} \Rightarrow 1 \right]$

where the probabilities are over the specified random choices and the coins used by  $\mathcal{A}$ . In the tcr game  $\mathcal{A}$  is actually a pair of algorithms  $\mathcal{A}_1$  and  $\mathcal{A}_2$ . Now letting  $F$  be an algorithm given oracle access to an ideal compression function  $f = \text{RF}_{k+n+d,n}$  we define the last security experiment:

- pro:  $\epsilon = \Pr \left[ K \stackrel{\$}{\leftarrow} \mathcal{K} : A_K^{f, f}(K) \Rightarrow 1 \right] - \Pr \left[ K \stackrel{\$}{\leftarrow} \mathcal{K} : A_K^{\mathcal{F}, S_K^{\mathcal{F}}}(K) \Rightarrow 1 \right]$

where the probabilities are over the specified random choices, the coins used by  $\mathcal{A}$  and  $\mathcal{S}$ , and the coins used by  $\mathcal{F} = \text{RF}_{\text{Dom}, \text{Rng}}$  and  $f = \text{RF}_{n+d, n}$ . The simulator  $\mathcal{S}$  maintains state across queries and has oracle access to  $\mathcal{F}$ . For more details on the pseudorandom oracle definition see [6, III.13].

We say that  $F$  is  $(t, L, \epsilon)$ -xxx for  $\text{xxx} \in \{\text{tcr}, \text{cr}\}$  if any adversary  $\mathcal{A}$  running in time at most  $t$  and outputting messages of length less than or equal to  $L$  bits has at most  $\epsilon$  probability of success in the xxx game. Similarly we say that  $F$  is a  $(t, q, L, \epsilon)$ -xxx for  $\text{xxx} \in \{\text{mac}, \text{prf}\}$  if any adversary  $\mathcal{A}$  running in time at most  $t$  and making at most  $q$  queries each of which has length at most  $L$  has at most  $\epsilon$  probability of success in the xxx game. Lastly we say that  $F$  is a  $(t_{\mathcal{A}}, t_{\mathcal{S}}, q_1, q_2, L, \epsilon)$ -pro if there exists a simulator  $\mathcal{S}$  running in time  $t_{\mathcal{S}}$  such that the following is true. Any adversary  $\mathcal{A}$  running in time at most  $t_{\mathcal{A}}$  and asking at most  $q_1$  ( $q_2$ ) queries to its first (second) oracle, with maximal query length  $L$  bits, has probability at most  $\epsilon$  of success in the pro game.

### 3 Hash Functions in the Dedicated Key Setting

**HASH FUNCTION HETEROGENEITY.** The first major benefit of dedicated-key hash functions is the enablement of *hash function heterogeneity*, which allows for the utilization of numerous different hash function instances. To understand why this is useful for security, we discuss (as an example) an important application of publicly-computable, collision-resistant hash functions: digital signature schemes. Recall that in such a scheme each party  $i$  picks a public key  $pk_i$  and publishes it. To verify a message, one hashes it and then applies some verification algorithm that utilizes  $pk_i$ . In current practice, all users utilize a single hash function  $H^h$ , for example SHA-1. Now that Wang, Yin, and Yu discovered a collision-outputting algorithm  $\mathcal{A}$  against  $H^h = \text{SHA-1}$  [26], simply running  $\mathcal{A}$  a single time compromises the security of *every* user's digital signature scheme.

If we instead utilize a dedicated-key hash function  $H^h: \mathcal{K} \times \mathcal{M} \rightarrow \{0, 1\}^n$  within our scheme, then each user  $i$  can pick a key  $K_i \in \mathcal{K}$  and publish it as part of their public key. In this way each user has his or her own hash function instance, exemplifying hash function heterogeneity. Now, attackers are faced with a significantly more difficult task, from a practical perspective. If they can construct a *categorical* attack algorithm  $\mathcal{A}$  (i.e., one that works equally well on any key), and if  $\mathcal{A}$  executes in  $w$  operations, then to attack a single user  $i$  requires (as before)  $w$  work. But attacking two users requires  $2w$  work, and in general attacking a group of  $p$  users requires  $pw$  work. If  $w \approx 2^{69}$ , as is the case for Wang, Yin, and Yu's SHA-1 attack [26], then even doubling the amount of work is a significant hurdle to mounting attacks in practice. The situation is even worse for the attackers if their attack algorithm is *key-specific* (i.e., it only works well on a particular key), because then they might have to adapt their attack to each user's key, which could require more cryptanalytic effort. In either case, hash function heterogeneity is a significant benefit of the dedicated-key setting, particularly when attacks are found that are just on the cusp of practicality.

IMPROVED SECURITY GUARANTEES. An important and wide-spread application of hash functions is for message authentication code (MAC) schemes, where we require hash functions to be unforgeable. To utilize a traditional hash function  $H^h: \mathcal{M} \rightarrow \{0, 1\}^n$  as a MAC scheme,  $H^h$  must be *keyed*, which means some of the input is set aside (a posteriori) for key bits. The canonical construct in this domain is HMAC [3,2], which is widely standardized and used. (NIST FIPS 198, ANSI X9.71, IETF RFC 2104, SSL, SSH, IPSEC, TLS, IEEE 802.11i, and IEEE 802.16e are only some instances.) Note that in these applications keys are secret and never revealed publicly.

In the traditional setting, the unforgeability of MACs built from hash functions requires the compression function to be a pseudorandom function (PRF), when keyed appropriately, and the transform to be PRF-Pr. However, unforgeability is a *weaker* security goal than being a PRF: any PRF is a good MAC but not vice versa. The reason we have to rely on PRFs for message authentication is that building transforms that preserve the unforgeability of a compression function  $h: \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  is inherently difficult and, in fact, no unforgeability preserving (which we'll call MAC-Pr) transforms are known in this setting.

On the other hand, if we work in the dedicated-key setting, then there are straightforward MAC-Pr transforms [1,15,14]. This allows us to utilize hash functions as MACs under just the assumption that  $h: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  is a good MAC, which provides a better security guarantee. To see why, note that an attack showing that  $h$  is not a PRF does *not* immediately imply that  $h$  can be forged against and therefore we can still have a guarantee that  $H^h$  is a secure MAC — but this is only true in the dedicated-key setting. In the prior setting we would lose all security guarantees.

Another benefit of the dedicated-key setting is that building transforms which are provably PRF-Pr becomes much easier. As we show in Section 5.3, *all* the transforms we consider are PRF-Pr, and the proofs of this are straightforward.

KEYING AND COLLISION-RESISTANCE. Hash functions with dedicated key inputs are an easy solution for the *foundations-of-hashing dilemma* [22], which is a problem of theoretical interest. The dilemma refers to the fact that  $h: \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$ , for  $d \neq 0$ , can *not* be collision-resistant: by the pigeonhole principle there are two distinct strings  $X, X' \in \{0, 1\}^{n+d}$  such that  $h(X) = h(X')$ . Thus there always exists an efficient collision-outputting algorithm  $\mathcal{A}$ , namely the one that outputs  $(X, X')$ . However, as Rogaway discusses at length in [22], rigorous provable security for keyless hash functions is still meaningful, since we can give explicit reductions (though at the cost of slightly more complex theorem statements). So while the dedicated-key setting enables formally meaningful collision-resistance and thus simpler theoretical treatments of CR hashing, the practical impact of this benefit is small.

EFFICIENCY. A significant downside of dedicated keys is efficiency loss. For every message block hashed using a dedicated-key compression function  $h: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$ , a total of  $k + n + d$  bits must be processed. Compare this to the situation of current popular hash functions, which only have to process  $n + d$  bits per block. The efficiency of the hash function therefore goes down by

about  $\frac{k}{n+d}$ , which could be an issue in settings where speed is paramount (e.g., message authentication of network traffic).

**BACKWARDS-COMPATIBILITY.** In many settings it will be desirable to utilize a hash function that does *not* reveal a dedicated-key input. This will be particularly true for backwards-compatibility with existing applications that utilize a standard hash function  $H: \{0, 1\}^* \rightarrow \{0, 1\}^n$ . We point out that it is easy to allow such compatibility when starting with a dedicated-key hash function  $H': \mathcal{K} \times \{0, 1\}^* \rightarrow \{0, 1\}^n$ . Simply fix an honestly generated and publicly-known key  $K$  (chosen, for example, by some trusted entity), and define the unkeyed hash function as  $H(M) = H'(K, M)$ .

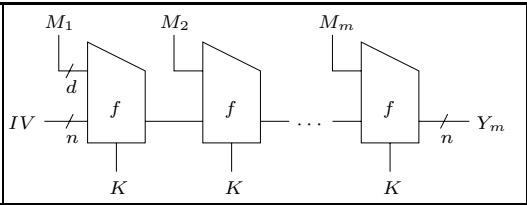
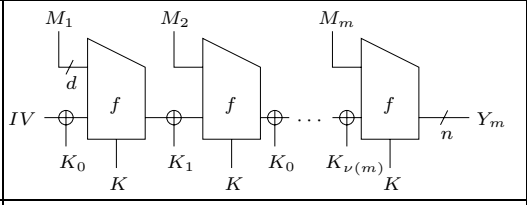
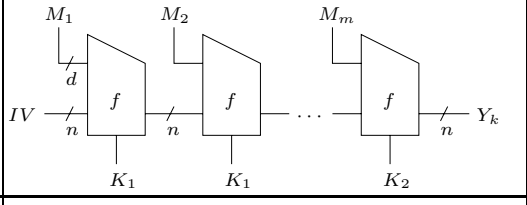
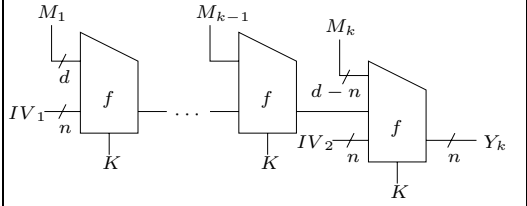
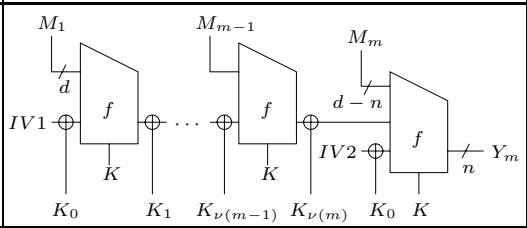
**ADVERSARIALLY-CHOSEN KEYS.** Most current cryptographically-sanctified applications of hash functions (e.g., digital signature schemes, message authentication codes, key derivation, and standard uses of random oracles) only require security for honestly generated dedicated keys. However, given the wide-spread use of hash functions in non-standard settings, one should be aware of the potential for abuse in applications that would require security even in the face of adversarially-chosen keys. A simple solution for such settings could be to require a fixed, honestly-generated key as mentioned above.

## 4 Dedicated Key Transforms

Let  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  be a dedicated-key compression function with  $d \geq n \geq 64$ . We now describe the various transforms treated in this paper. A transform  $H$  describes how to utilize  $f$  (as a black box) in order to generate a hash function  $H^f: \mathcal{K} \times \mathcal{M} \rightarrow \{0, 1\}^n$ . A transform is defined in two separate steps. We first specify an injective padding function that maps from  $\{0, 1\}^*$  or  $\{0, 1\}^{\leq 2^{64}}$  to either  $D^+ = \cup_{i \geq 1} \{0, 1\}^{id}$  or  $D^\circ = \cup_{i \geq 1} \{0, 1\}^{id+d-n}$ . Then we specify an iteration function which describes how to hash strings in either  $D^+$  or  $D^\circ$ . We define the following padding functions:

- **pad:**  $\{0, 1\}^* \rightarrow D^+$  is defined by  $\text{pad}(M) = M \parallel 10^r$
- **pad<sub>s</sub>:**  $\mathcal{D} \rightarrow D^+$  is defined by  $\text{pad}_s(M) = M \parallel 10^r \parallel \langle |M| \rangle_{64}$
- **padPF:**  $\{0, 1\}^* \rightarrow D^+$  is a prefix-free padding function: for any  $M, M' \in \{0, 1\}^*$  where  $|M| < |M'|$  we have that  $\text{padPF}(M)$  is not a prefix of  $\text{padPF}(M')$ . (For example: pad to make the length a multiple of  $d - 1$ , parse the result into blocks of  $d - 1$  bits, append a zero to each block except the final block, and append one to the final block.)
- **padCS:**  $\{0, 1\}^* \rightarrow D^\circ$  is defined by  $\text{padCS}(M) = M \parallel 10^r$
- **padCS<sub>s</sub>:**  $\mathcal{D} \rightarrow D^\circ$  is defined by  $\text{padCS}_s(M) = M \parallel 10^r \parallel \langle |M| \rangle_{64} \parallel 0^p$

where for **pad**, **pad<sub>s</sub>**, and **padCS** the value  $r$  is the minimal number of zeros so that the returned string is in the range of the padding function. For **padCS<sub>s</sub>** we define  $p$  in two potential ways. If  $d \geq n + 64$  (there is room for the strengthening in the envelope), then  $p = 0$ . If  $d < n + 64$  (there is not enough room for the strengthening in the envelope), then  $p = d - n$ . Then  $r$  is the number of zeros

|   |   |
|---|---|
| <p><b>Algorithm <math>f^+(K, M)</math>:</b><br/> <math>M_1 \cdots M_m \stackrel{d}{\leftarrow} M</math>; <math>Y_0 \leftarrow IV</math><br/> <b>for</b> <math>i = 1</math> <b>to</b> <math>m</math> <b>do</b><br/> <math>Y_i \leftarrow f_K(Y_{i-1} \parallel M_i)</math><br/> <b>ret</b> <math>Y_m</math></p>  |   |
| <p><b>Algorithm <math>f^{\text{Sh}}((K, \{K_i\}_1^t), M)</math>:</b><br/> <math>M_1 \cdots M_m \stackrel{d}{\leftarrow} M</math>; <math>Y_0 \leftarrow IV</math><br/> <b>for</b> <math>i = 1</math> <b>to</b> <math>m</math> <b>do</b><br/> <math>Y_i \leftarrow f_K((Y_{i-1} \oplus K_{\nu(i)}) \parallel M_i)</math><br/> <b>ret</b> <math>Y_m</math></p>                           |   |
| <p><b>Algorithm <math>f^{\text{NI}}((K_1, K_2), M)</math>:</b><br/> <math>M_1 \cdots M_m \stackrel{d}{\leftarrow} M</math><br/> <math>Y_{m-1} \leftarrow f^+(K_1, M_1 \cdots M_{m-1})</math><br/> <b>ret</b> <math>Y_m \leftarrow f_{K_2}(Y_{m-1} \parallel M_m)</math></p>   |   |
| <p><b>Algorithm <math>f^{\text{CS}}(K, M)</math>:</b><br/> <math>M_1 \cdots M_m \stackrel{d}{\leftarrow} M</math><br/> <math>Y_{m-1} \leftarrow f^+(K, M_1 \cdots M_{m-1})</math><br/> <b>ret</b> <math>f_K(IV2 \parallel Y_{m-1} \parallel M_m)</math></p>   |   |
| <p><b>Algorithm <math>f^{\text{ESh}}((K, \{K_i\}_1^t), M)</math>:</b><br/> <math>M_1 \cdots M_m \stackrel{d}{\leftarrow} M</math><br/> <math>Y \leftarrow f^{\text{Sh}}((K, \{K_i\}_1^t), M_1 \cdots M_{m-1})</math><br/> <math>I \leftarrow IV2 \oplus K_0</math>; <math>Y' \leftarrow Y \oplus K_{\nu(m)}</math><br/> <b>ret</b> <math>f_K(I \parallel Y' \parallel M_m)</math></p> |  |

**Fig. 2.** The algorithms and diagrams detailing the iteration functions we consider. Transforms are the composition of an iteration function and a padding function.

needed to make the returned string in  $D^\circ$ . Note that we restrict our attention to padding functions  $g$  such that for any messages  $M, M'$  for which  $|M| = |M'|$  we have that  $|g(M)| = |g(M')|$ .

The iteration functions we consider are specified in Figure 2, and we use them to now define the seven previously proposed transforms. The basic Merkle-Damgård (MD) iteration  $f^+$ :  $\{0, 1\}^k \times D^+ \rightarrow \{0, 1\}^n$  repeatedly applies  $f$ . The **plain** MD transform [16, 12] is  $\text{MD}[f] = f^+(k, \text{pad}(M))$ . The **strengthened** MD transform [12] is  $\text{sMD}[f] = f^+(k, \text{pad}_s(M))$ . The **prefix-free** MD

transform [15] is  $\text{Pre}[f] = f^+(\kappa, \text{padPF}(M))$ . The placeholders  $\kappa$  and  $M$  designate how to handle the key and message inputs. The Shoup iteration  $f^{\text{Sh}}: (\{0, 1\}^k \times \{0, 1\}^{tn}) \times D^+ \rightarrow \{0, 1\}^n$  utilizes  $t = \lceil \log_2(\sigma) \rceil$  key masks where  $\sigma$  the maximal number of iterations of  $f$  allowed. Also we define  $\nu(i)$  to be the largest value  $x$  such that  $2^x$  divides  $i$ . The key masks  $\{K_i\}_1^t \in \{0, 1\}^{tn}$  are  $t$   $n$ -bit keys that are used to ‘mask’ chaining variables with secret key material. Then the **Shoup** transform [24] is  $\text{Sh}[f] = f^{\text{Sh}}(\kappa, \text{pad}_s(M))$ . The nested iteration  $f^{\text{NI}}: (\{0, 1\}^k \times \{0, 1\}^k) \times D^+ \rightarrow \{0, 1\}^n$  just envelopes an  $f^+$  iteration with an application of  $f$  using a second key. The **strengthened Nested Iteration** transform [1] is  $\text{sNI}[f] = f^{\text{NI}}(\kappa, \text{pad}_s(M))$ . The **Nested Iteration** transform [15] is  $\text{NI}[f] = f^{\text{NI}}(\kappa, \text{pad}(M))$ . The chain shift iteration  $f^{\text{CS}}: \{0, 1\}^k \times D^o \rightarrow \{0, 1\}^n$ , envelopes an internal  $f^+$  iteration with an application of  $f(\text{IV2} \parallel \cdot)$ . We require that  $\text{IV2} \neq \text{IV1}$ . The **chain shift** transform [15] is  $\text{CS}[f] = f^{\text{CS}}(\kappa, \text{padCS}(M))$ .

Now we define two new transforms. The **strengthened Chain Shift** transform  $\text{sCS}[f] = f^{\text{CS}}(\kappa, \text{padCS}_s(M))$  adds strengthening to the CS transform. The enveloped Shoup iteration  $f^{\text{ESh}}: (\{0, 1\}^k \times \{0, 1\}^{tn}) \times D^o \rightarrow \{0, 1\}^n$  uses an internal  $f^{\text{Sh}}$  iteration and then an envelope like the one used in  $f^{\text{CS}}$ . Note that the output of  $f^{\text{Sh}}$  is xor’d with a key mask and  $\text{IV2}$  is xor’d with  $K_0$  (which serves to preserve that  $\text{IV1} \neq \text{IV2}$  across the masking). Then the **Enveloped Shoup** transform is  $\text{ESh}[f] = f^{\text{ESh}}(\kappa, \text{padCS}_s(M))$ .

For a fixed compression function  $f$  each transform defines a hash function, e.g.  $\text{MD}^f = \text{MD}[f]$  is the hash function with signature  $\text{MD}^f: \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^n$  defined by  $\text{MD}^f(K, M) = f^+(K, \text{pad}(M))$ .

For each padding function  $g$  (and therefore each transform) we define an *efficiency function*  $\tau_g: \mathbb{N} \rightarrow \mathbb{N}$  defined as  $\tau_g(L) = \lceil |g(M)|/d \rceil$  for any  $M \in \{0, 1\}^L$ . For brevity we will often just write  $\tau(L)$  where the padding function of interest is clear from context. Note that efficiency functions are called application functions in [15]. The efficiency functions are given in Figure 1, where for  $\text{padPF}$  we utilize the concrete example outlined above.

## 5 Security Analysis of the Transforms

We now analyze the security of the nine transforms in terms of the five different security goals. The summary of our analysis is given in Figure 1. Some of the results are already established by prior work; we omit discussion of these and refer the reader to the given citations. We discuss each security property in turn. Due to a lack of space, we cannot include any proofs here. A complete treatment, including proofs, is given in the full version of the paper [5].

### 5.1 Collision Resistance Preservation

Collision-resistance preservation is typically achieved via strengthening: appending the length of a message to it before processing. Not surprisingly, transforms that omit strengthening are *not* CR-Pr: we show this for Pre, NI, and CS. On the other hand, those that include strengthening are CR-Pr, as we show for sNI, sCS, and ESh.

**Theorem 1.** [Negative results: Pre, NI, CS] *Let  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^{n-1}$  be a  $(t, n + d, \epsilon)$ -cr function. Then there exists a function  $g: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  that is  $(t - c_1, n + d, \epsilon)$ -cr but  $\text{Pre}[g], \text{NI}[g], \text{CS}[g]$  are at most  $(c_2, 3d, 1)$ -cr where  $c_1, c_2$  are small constants.  $\square$*

**Theorem 2.** [Positive results: sNI, sCS, ESh] *Let  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  be a  $(t, n + d, \epsilon)$ -cr function. Then  $\mathbb{T}[f]$  is  $(t', L, \epsilon')$ -cr where for*

- (1)  $\mathbb{T} = \text{sNI}$  we have  $t' = t - c_1\tau(L)$  and  $\epsilon' = \epsilon/2$
- (2)  $\mathbb{T} \in \{\text{sCS}, \text{ESh}\}$  we have  $t' = t - c_2\tau(L)$  and  $\epsilon' = \epsilon$

where  $c_1, c_2$  are small constants.  $\square$

### 5.2 MAC (Unforgeability) Preservation

We show that MD, sMD, and Sh *do not* preserve unforgeability. Recall that in this setting, the key material (including the key masks of Sh) is secret and therefore unknown to the adversary. While it may not be surprising that MD and sMD are not MAC-Pr, one might be tempted to think that the large amount of secret key material used in Sh could assist in preserving unforgeability. Unfortunately this is not the case, though the counter-example is involved [5]. On the positive side, we have that ESh is MAC-Pr, which can be shown using the proof techniques in [15].

**Theorem 3.** [Negative results: MD, sMD, Sh] *If there exists a function  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^{n-1}$  that is a  $(t, q, n + d, \epsilon)$ -mac, then there exists a function  $g: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  that is a  $(t - c_1q, q, n + d, \epsilon)$ -mac but*

- (1)  $\text{MD}[g], \text{sMD}[f']$  are at most  $(c_2, n - 1, 3d, 1/2)$ -mac
- (2)  $\text{Sh}[g]$  is at most a  $(c_3, 2(n - 1), 3d, 1/4)$ -mac

where  $c_1, c_2,$  and  $c_3$  are small constants.  $\square$

**Theorem 4.** [Positive results: ESh] *Let  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  be a  $(t, q, \epsilon)$ -mac. Then  $\text{ESh}[f]$  is a  $(t - c(q' + 1)\tau(L), q', L, \epsilon')$ -mac where*

$$q' = (q - \tau(L) + 1)/\tau(L) \quad \text{and} \quad \epsilon' = (q^2/2 + 3q/2 + 1)\epsilon$$

for  $c$  a small constant and any  $\{K_i\}_1^t \in \{0, 1\}^{tn}$  with  $t = \log_2(\tau(L))$ .  $\square$

### 5.3 Pseudorandom Function Preservation

In the non-dedicated-key setting, building PRF preserving transforms is non-trivial and the proofs of security are complex. In stark contrast to this, we show that all of the dedicated-key transforms considered here are PRF-Pr, and the proofs establishing this are relatively straightforward (see [5]).

**Theorem 5.** [Positive results: MD, sMD, Pre, Sh, sNI, NI, CS, sCS, ESh] *Let  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  be a  $(t, q, n + d, \epsilon)$ -prf. Then  $\mathbb{T}[f]$  is a  $(t', q', L, \epsilon')$ -prf where for*

- (1)  $\mathbb{T} \in \{\text{MD}, \text{sMD}, \text{Pre}, \text{Sh}, \text{CS}, \text{sCS}, \text{ESh}\}, t' = t - cq\tau(L), q' = q/\tau(L), \epsilon' = \epsilon + q^2\tau(L)^2/2^n,$
  - (2)  $\mathbb{T} \in \{\text{sNI}, \text{NI}\}, t' = t - cq\tau(L), q' = q/\tau(L), \epsilon' = 2\epsilon + q^2(\tau(L) - 1)^2/2^n$
- where  $c$  is a small constant and  $\{K_i\}_1^t \in \{0, 1\}^{tn}$  for  $t = \log_2(\tau(L))$ .  $\square$



## 5.4 Pseudorandom Oracle Preservation

Establishing that a transform is PRO-Pr ensures that a hash function constructed with it “behaves like a random oracle” under the assumption that the compression function is ideal. This is important for usage of hash functions to instantiate random oracles, as discussed at length in [11]. To reason about dedicated-key transforms, we model a compression function  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  as a family of random oracles, one for each key in  $\{0, 1\}^k$ . However, since we only ever use one or two fixed keys from  $\{0, 1\}^k$ , we will (without loss of generality) simply utilize two separate random oracles  $f = \text{RF}_{n+d,n}$  and  $g = \text{RF}_{n+d,n}$  from the family. This simplifies our analysis, and, in particular, means that many results from the keyless setting carry over directly to the dedicated-key setting (see Figure 1). For example, the negative results that  $\text{MD}^f$  and  $\text{sMD}^f$  are not PROs follows from simple length-extension attacks (see [11]). The security of CS, and sCS is implied by the security of EMD [6] and the security of sNI and NI is implied by results in [11].

We point out that  $\text{Sh}^f$  is not a PRO. Since the key masks are public, simple length extension attacks enable an adversary to differentiate between it and a true variable-input-length random oracle. On the other hand  $\text{ESh}^f$  is a PRO.

**Theorem 6. [Negative result: Sh]** *Let  $f = \text{RF}_{n+d,n}$  be a random oracle. Then there exists an  $(c, t_S, 2, 1, 2d, 1 - 2^{-n})$ -pro adversary  $\mathcal{A}$  against  $\text{Sh}^f$  for any simulator  $\mathcal{S}$  with arbitrary running time  $t_S$ . The running time of  $\mathcal{A}$  is a small constant  $c$ .  $\square$*

**Theorem 7. [Positive result: ESh]** *Let  $f = \text{RF}_{n+d,n}$  be a random oracle. Then  $\text{ESh}^f$  is a  $(t_A, t_S, q_1, (q_2 + q_3), L, \epsilon)$ -pro where*

$$\epsilon \leq (l^2 q_1^2 + (l q_1 + q_2)(q_2 + q_3))/2^n + l q_1/2^n$$

for  $l = \tau(L)$ . The running time  $t_A$  is arbitrary while  $t_S = \mathcal{O}(q_2^2 + q_2 q_3)$ .  $\square$

## 5.5 Target Collision Resistance Preservation

Universal one-way hash functions (UOWHF) were first introduced by Naor and Reingold [18] and later went by the term target collision resistance (TCR) [10]. The best TCR-Pr transforms known require a logarithmic (in the size of the messages allowed) amount of key material. Mironov has given strong evidence that one cannot do better [17]. Due to this and because any CR function is also TCR [23], it might be sufficient in most settings to utilize a dedicated-key transform that preserves the four previous properties. Still, target-collision resistant functions are useful in some settings [10] and establishing their security based only on the compression function being TCR results in a stronger guarantee of security (this is analogous to the discussion of MAC-Pr versus PRF-Pr in Section 3). Thus, we extend our analysis to this property.

In light of Mironov’s result, it is not surprising that Pre, sNI, NI, CS, and sCS are not TCR-Pr, though we establish these facts directly. On the other hand, a

proof that ESh is TCR-Pr can be straightforwardly derived from the proof that Sh is TCR (see [17] or [24]).

**Theorem 8. [Negative results: Pre, sNI, NI, CS, sCS]** *If there exists a function  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^{n-1}$  that is  $(t, n + d, \epsilon)$ -tcr, then there exists a function  $g: \{0, 1\}^k \times \{0, 1\}^{(n+d)} \rightarrow \{0, 1\}^n$  that is  $(t - c_1, n + d, \epsilon + 2^{-k+1})$ -tcr but  $\text{Pre}[g], \text{sNI}[g], \text{NI}[g], \text{CS}[g], \text{sCS}[g]$  are at most  $(c_2, 3d, 1 - 2^{-k})$ -tcr where  $c_1, c_2$  are small constants.  $\square$*

**Theorem 9. [Positive result: ESh]** *Let  $f: \{0, 1\}^k \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^n$  be  $(t, n + d, \epsilon)$ -tcr. Then  $\text{ESh}[f]$  is  $(t - ct_f^2 \tau(L)^2, L, \epsilon \tau(L))$ -tcr for a small constant  $c$  and where  $T_f$  is the time to execute  $f$ .  $\square$*

## Acknowledgments

The authors are supported in part by NSF grants CNS 0524765, CNS 0627779, and a gift from Intel Corporation.

## References

1. An, J., Bellare, M.: Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 252–269. Springer, Heidelberg (1999)
2. Bellare, M.: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 113–120. Springer, Heidelberg (2006)
3. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
4. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: the cascade construction and its concrete security. In: Proceedings of the 37th Annual Symposium on Foundations of Computer Science – FOCS '96, IEEE Computer Society, pp. 514–523. IEEE Computer Society Press, Los Alamitos (1996)
5. Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms (2007), Full version of current paper, <http://www.cse.ucsd.edu/users/mihir/>
6. Bellare, M., Ristenpart, T.: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
7. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. First ACM Conference on Computer and Communications Security – CCS '93, pp. 62–73. ACM Press, New York (1993)
8. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
9. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)

10. Bellare, M., Rogaway, P.: Collision-Resistant Hashing: Towards Making UOWHFs Practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
11. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 21–39. Springer, Heidelberg (2005)
12. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
13. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
14. Maurer, U., Sjödin, J.: Domain Expansion of MACs: Alternative Uses of the FIL-MAC. In: Smart, N.P. (ed.) Cryptography and Coding. LNCS, vol. 3796, pp. 168–185. Springer, Heidelberg (2005)
15. Maurer, U., Sjödin, J.: Single-key AIL-MACs from any FIL-MAC. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 472–484. Springer, Heidelberg (2005)
16. Merkle, R.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
17. Mironov, I.: Hash functions: from Merkle-Damgård to Shoup. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 166–181. Springer, Heidelberg (2001)
18. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing – STOC’89, pp. 33–43. ACM Press, New York (1989)
19. National Institute of Standards and Technology. FIPS PUB 180-1: Secure Hash Standard. Supersedes FIPS PUB 180 1993 May 11 (1995)
20. RSA Laboratories. RSA PKCS #1 v2.1: RSA Cryptography Standards (2002)
21. Rivest, R.: The MD4 Message Digest Algorithm. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
22. Rogaway, P.: Formalizing Human Ignorance: Collision-Resistant Hashing without the Keys. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 221–228. Springer, Heidelberg (2006)
23. Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
24. Shoup, V.: A Composition Theorem for Universal One-Way Hash Functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
25. Tsudik, G.: Message Authentication with One-way Hash Functions. SIGCOMM Comp. Commun. Rev. 22(5), 29–38 (1992)
26. Wang, X., Yin, Y., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
27. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

# Unrestricted Aggregate Signatures

Mihir Bellare<sup>1</sup>, Chanathip Namprempre<sup>2</sup>, and Gregory Neven<sup>3</sup>

<sup>1</sup> Department of Computer Science & Engineering,  
University of California San Diego, La Jolla, CA 92093, USA  
mihir@cs.ucsd.edu

<http://www-cse.ucsd.edu/users/mihir>

<sup>2</sup> Electrical Engineering Department, Faculty of Engineering,  
Thammasat University, Klong Lueng, Patumtani 12120, Thailand  
nchanath@engr.tu.ac.th

<http://chanathip.ee.engr.tu.ac.th>

<sup>3</sup> Department of Electrical Engineering, Katholieke Universiteit Leuven,  
Kasteelpark Arenberg 10, B-3001 Heverlee-Leuven, Belgium  
Gregory.Neven@esat.kuleuven.be  
<http://www.neven.org>

**Abstract.** Secure use of the BGLS [1] aggregate signature schemes is restricted to the aggregation of distinct messages (for the basic scheme) or per-signer distinct messages (for the enhanced, prepend-public-key version of the scheme). We argue that these restrictions preclude interesting applications, make usage of the schemes error-prone and are generally undesirable in practice. Via a new analysis and proof, we show how the restrictions can be lifted, yielding the first truly unrestricted aggregate signature scheme. Via another new analysis and proof, we show that the distinct signer restriction on the sequential aggregate signature schemes of [2] can also be dropped, yielding an unrestricted sequential aggregate signature scheme.

## 1 Introduction

AGGREGATE SIGNATURES. An aggregate signature (AS) scheme [1] is a digital signature scheme with the additional property that a sequence  $\sigma_1, \dots, \sigma_n$  of individual signatures —here  $\sigma_i$  is the signature, under the underlying base signature scheme, of some message  $m_i$  under some public key  $pk_i$ — can be condensed into a single, compact aggregate signature  $\sigma$  that simultaneously validates the fact that  $m_i$  has been signed under  $pk_i$  for all  $i = 1, \dots, n$ . There is a corresponding aggregate verification process that takes input  $(pk_1, m_1), \dots, (pk_n, m_n), \sigma$  and accepts or rejects. Aggregation is useful to reduce bandwidth and storage, and is especially attractive for mobile devices like sensors, cell phones, and PDAs where communication is more power-expensive than computation and contributes significantly to reducing battery life.

SCHEMES. Boneh, Gentry, Lynn, and Shacham [1] present an aggregate signature scheme based on the BLS signature scheme [3]. We call it *AS-1* and represent

it succinctly in the first row of Table [1](#)  $\mathcal{AS}\text{-}1$ , however, has some limitations. As the table shows, the aggregate verification process, on inputs  $(pk_1, m_1), \dots, (pk_n, m_n), \sigma$ , rejects if the messages  $m_1, \dots, m_n$  are not distinct. The restriction is crucial because, without it, as shown in [1](#), the scheme is subject to a forgery attack. The consequence, however, is to restrict the ability to aggregate to settings where the messages signed by the signers are all different. BGLS recognize this limitation and suggest a workaround. Specifically, they say: “It is easy to ensure the messages are distinct: The signer simply prepends her public key to every message she signs ...” [[1](#) Section 3.2]. They stop short of specifying a scheme in full, but since it is clearly their intention to “reduce to the previous case,” our understanding is that they are advocating the modified scheme in which the signature of a message  $m$  under  $pk$  is the BLS signature of the *enhanced message*  $M = pk\|m$  under  $pk$  while aggregate verification is done by applying the aggregate verification procedure of  $\mathcal{AS}\text{-}1$  to  $(pk_1, pk_1\|m_1), \dots, (pk_n, pk_n\|m_n), \sigma$ . However, if so, in this scheme, which we call  $\mathcal{AS}\text{-}2$ , the aggregate verification process will reject unless the enhanced messages  $pk_1\|m_1, \dots, pk_n\|m_n$  are distinct. (Why? Because the aggregate verification process of  $\mathcal{AS}\text{-}1$  rejects unless the messages are distinct, and the role of the messages is now played by the enhanced messages.) The consequence is that the ability to aggregate is restricted to settings where the enhanced messages signed by the signers are all different. That is, the limitations have been reduced, but not eliminated.

**OUR RESULT.** We ask whether there exists a truly unrestricted proven-secure aggregate signature scheme. Namely, there should be no distinctness-based restriction of any kind, whether on messages or enhanced messages. We show that the answer is yes. Our result is a new, direct analysis of the security of enhanced-message signature aggregation which shows that the distinctness condition in the aggregate verification process of  $\mathcal{AS}\text{-}2$  —namely that this process rejects if any two enhanced messages are the same— can be dropped without compromising security. In other words, an unrestricted scheme can be obtained by the natural adaptation of  $\mathcal{AS}\text{-}2$  in which the distinctness condition in the verification is simply removed and all else is the same. This scheme, which we denote  $\mathcal{AS}\text{-}3$ , is summarized in the last row of Table [1](#). The fact that  $\mathcal{AS}\text{-}3$  is very close to  $\mathcal{AS}\text{-}2$  is a plus because it means existing implementations can be easily patched.

We clarify that the security of  $\mathcal{AS}\text{-}3$  is not proved in [1](#). They prove secure only  $\mathcal{AS}\text{-}1$ . The security of  $\mathcal{AS}\text{-}2$  is a consequence, but the security of  $\mathcal{AS}\text{-}3$  is not. What we do instead is to *directly* analyze security in the case that signatures are on enhanced messages. Our analysis explicitly uses and exploits the presence of the prepended public keys to obtain the stronger conclusion that  $\mathcal{AS}\text{-}3$  (not just  $\mathcal{AS}\text{-}2$ ) is secure.

**MOTIVATION.** The limitation of  $\mathcal{AS}\text{-}2$  —namely that aggregation is restricted to settings where no two enhanced messages are the same— may seem minor, because all it says is that a set of signatures to be aggregated should not contain duplicates, meaning multiple signatures by a particular signer of a particular message. However, as we now explain, there are several motivations for schemes like  $\mathcal{AS}\text{-}3$  where aggregation is possible even in the presence of duplicates.

**Table 1.** The aggregate signature schemes we discuss. Here  $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  is a bilinear map,  $g$  is a generator of  $\mathbb{G}_2$  known to all parties, and  $H: \{0, 1\}^* \rightarrow \mathbb{G}_1$  is a hash function. The second column shows the signature of a message  $m$  under public key  $g^x$ , generated using secret key  $x$ . In all cases, a sequence of signatures is aggregated by simply multiplying them in  $\mathbb{G}_1$ . The third column shows under what conditions the aggregate verification algorithm accepts  $\sigma$  as a valid aggregate signature of messages  $m_1, \dots, m_n$  under public keys  $g^{x_1}, \dots, g^{x_n}$  respectively.

| Scheme                                      | Sign            | Aggregate verification process accepts iff  |
|---|-----------------|---|
| $\mathcal{AS}\text{-}1$ <a href="#">[1]</a> | $H(m)^x$        | $e(\sigma, g) = \prod_{i=1}^n e(H(m_i), g^{x_i})$ and $m_1, \dots, m_n$ all distinct                                  |
| $\mathcal{AS}\text{-}2$ <a href="#">[1]</a> | $H(g^x \  m)^x$ | $e(\sigma, g) = \prod_{i=1}^n e(H(g^{x_i} \  m_i), g^{x_i})$ and $g^{x_1} \  m_1, \dots, g^{x_n} \  m_n$ all distinct |
| $\mathcal{AS}\text{-}3$                     | $H(g^x \  m)^x$ | $e(\sigma, g) = \prod_{i=1}^n e(H(g^{x_i} \  m_i), g^{x_i})$  |

Consider a sensor network deployed in a remote area, for example, an environment monitoring network such as the tsunami early warning system already in operation in the Indian Ocean [\[4\]](#). The sensors periodically record measurements from the environment and send them to a monitoring center. In applications where integrity is important (say, to prevent attackers from raising a false alarm that a tsunami is coming), each sensor node must sign its data. The center aggregates these data and the signatures to save storage. (The schemes discussed here permit on-line aggregation in which one can maintain a current aggregate and aggregate into it a received signature.) However, environmental measurements can certainly repeat over time! Indeed, especially in stable conditions, we would expect frequent repeats. Thus, a single signer (sensor) may sign the same data (measurement) many times. In general, whenever the data being signed is drawn from a small space, not just messages, but even enhanced messages can repeat and an unrestricted aggregate signature scheme is necessary.

Perhaps an even more important reason to prefer unrestricted schemes is that they are less likely to be misused or to result in unexpected errors. An application designer contemplating using  $\mathcal{AS}\text{-}2$  must ask herself whether, in her application, enhanced messages might repeat. This may not only be hard to know in advance, but might also change with time. (Experience has repeatedly shown that once a piece of cryptography is deployed, it is used for purposes or applications beyond those originally envisaged.) With an unrestricted scheme, the application designer is freed from the need to worry about whether the setting of her application meets the restrictions, reducing the chance of error. In general, application designers and programmers have a hard enough time as it is to make error-free use of cryptography. Asking them to understand message distinctness restrictions and anticipate whether their application meets them is an added burden and one more place where things can go wrong.

POSSIBLE WORKAROUNDS. Various ways to get around message distinctness restrictions may come to mind, but these workarounds either do not work or tend

to be more complex or restrictive than direct use of an unrestricted scheme. For example, one could have the verifier drop from its input list  $(pk_1, m_1), \dots, (pk_n, m_n), \sigma$  any pair  $pk_i, m_i$  that occurred previously in the list, but then, assuming  $\sigma$  was a correct aggregate signature of the given list, it will not be a correct aggregate signature for the truncated list, and verification will fail. Another possibility is that the aggregator refrain from including in the aggregate any signature corresponding to a public key and message whose signature is already in the aggregate. But aggregation may be done on-line, and the aggregator may know only the current aggregate and the incoming signature, and have no way to tell whether or not it is a duplicate. Nonces (timestamps or sequence numbers) could be added to messages to render them unique, but this would complicate the application and increase cost.<sup>1</sup> Being able to use  $\mathcal{AS}\text{-}3$  without any worries about signature or message replication is simpler, easier, and more practical.

RESULTS FOR SASS. A sequential aggregate signature (SAS) scheme [2] permits a more restrictive kind of aggregation in which the signers must participate in the process and use their secret keys. Imagine the signers forming a chain. In step  $i$ , the  $i$ -th signer receives from the previous one a current aggregate, and, using its secret key, it aggregates into this its own signature, passing the new aggregate on to the next signer in the chain. The output of the final signer is the aggregate signature. Although clearly less powerful than general aggregate signature (GAS) schemes—following [5] we now use this term for the BGLS-type aggregate signatures discussed above in order to distinguish them from SASS—the argument of [2] is that sequential aggregation is possible for base signature schemes other than BLS or may be done more cheaply. Specifically, Lysyanskaya, Micali, Reyzin, and Shacham [2] present a SAS scheme based on certified [6] claw-free trapdoor permutations.

However, the model and schemes of [2] also have some limitations. They require that no public key can appear more than once in a chain. That is, the signers who are part of a signing chain must be distinct. But in practice there are many reasons to prefer to allow aggregation even when a particular signer signs multiple messages, meaning when there are loops in the chain and public keys can repeat. Certainly, the previously discussed motivations are still true. Namely, in applications like signing in sensor nets or ad hoc networks, a particular signer (sensor) will be producing many signatures and it would be convenient to allow these to be aggregated. More importantly, an unrestricted SAS scheme is more misuse resistant because it does not require the application designer to try to predict in advance whether or not there will be repeated signers in a prospective chain. But in fact the restrictions in [2] are even greater than the ones for  $\mathcal{AS}\text{-}2$ , for they say not only that one cannot aggregate when a particular signer signs a particular message twice, but that one cannot aggregate even when

<sup>1</sup> Nonces might be added anyway to prevent replay attacks. If so, well and good. But replay attacks are not always a concern. In some settings, an adversary might be able to inject bogus data, yet be unable to eavesdrop. Without eavesdropping, the adversary clearly cannot replay data. This could be true for the ocean sensor scenario we discussed above.

a particular signer signs two or more messages that are distinct. This makes the number of excluded applications even larger, for it is common that a particular signer needs to sign multiple messages.

Our result —analogous to the GAS case— is a new analysis and proof that shows that the restrictions imposed by [2] on their schemes can be lifted without impacting security (that is, verification can just drop the condition that one reject if there are repeated public keys, and security is preserved), yielding unrestricted schemes. Again, that only a minor modification to the scheme is needed is convenient if existing implementations need to be updated. The precise result statement and the accompanying proof are in the full version of this paper [7].

**TIGHT REDUCTION RESULTS.** The security of  $\mathcal{AS}\text{-}1$ ,  $\mathcal{AS}\text{-}2$ , and  $\mathcal{AS}\text{-}3$  is based on the coCDH problem. However, none of the security reductions —namely, those of [1] in the first two cases and ours in the last— are tight. By applying the technique of Katz and Wang [8], we obtain an alternative scheme  $\mathcal{AS}\text{-}4$  with a tight security reduction to coCDH. When implemented over the same group,  $\mathcal{AS}\text{-}1$ ,  $\mathcal{AS}\text{-}2$ ,  $\mathcal{AS}\text{-}3$ , and  $\mathcal{AS}\text{-}4$  all have about the same computational cost, the price for  $\mathcal{AS}\text{-}4$  being a slightly larger aggregate signature. (Specifically, one needs one extra bit per constituent signature.) However, due to the tight reduction,  $\mathcal{AS}\text{-}4$  is actually *more* efficient than the other schemes when one compares them at the same (provable) security level. (Because to achieve a given level of security,  $\mathcal{AS}\text{-}4$  needs a smaller group than the other schemes.) We also obtain an analogous result for the SAS case. The statements and the proofs of our tight reduction results are in the full version of this paper [7].

**RELATED WORK.** Lu, Ostrovsky, Sahai, Shacham, and Waters [9] present a SAS scheme for which they can lift the distinct signer restriction, as follows. If a signer wishes to add its signature of a message  $M_{\text{new}}$  to an aggregate  $S$  which already contains its signature  $S_{\text{old}}$  on some message  $M_{\text{old}}$ , then it removes  $S_{\text{old}}$  from  $S$  and then adds back in a signature on the message  $M_{\text{new}}\|M_{\text{old}}$ . However, their scheme is weaker than the others we have discussed —ours or those of [12]— with regard to some security properties and also with regard to efficiency. Specifically, [9] uses the certified public key model [10,11], which reflects the assumption that signers provide strong ZK proofs of knowledge of secret keys to the CA at key-registration, an assumption that we, following [12], do not make. Also, in the scheme of [9], public keys are very large, namely 162 group elements, which is particularly expensive if public keys have to be transmitted along with signatures. In contrast, other schemes have short public keys. On the other hand, the proofs of [9] are in the standard model, while ours, following [12], are in the random oracle model of [12].

Interestingly, in a survey paper, Boneh, Gentry, Lynn, and Shacham [5] present  $\mathcal{AS}\text{-}3$ , claiming that the results of [1] prove it secure. However, this appears to be an oversight because, as we have seen, the results of [1] prove  $\mathcal{AS}\text{-}2$  secure, not  $\mathcal{AS}\text{-}3$ . By proving  $\mathcal{AS}\text{-}3$  secure via our direct analysis, we are filling the gap and validating the claim of [5]. Shacham’s Ph.D thesis [13] notes that the concrete security of the reduction of [1] can be slightly improved by replacing messages with enhanced ones, but he does not claim security of  $\mathcal{AS}\text{-}3$ .



## 2 Notation and Basic Definitions

NOTATION AND CONVENTIONS. If  $x$  is a string, then  $|x|$  is the length of  $x$ . We denote by  $x_1 \| \cdots \| x_n$  an encoding of objects  $x_1, \dots, x_n$  as a binary string from which the constituent objects are uniquely recoverable. When the objects can be encoded as strings whose length is known from the context, simple concatenation will serve the purpose. If  $S$  is a finite set, then  $|S|$  is its size, and  $s \stackrel{\$}{\leftarrow} S$  means that  $s$  is chosen at random from  $S$ . We let  $e$  denote the base of the natural logarithm. An algorithm may be randomized unless otherwise indicated. An adversary is an algorithm. If  $A$  is an algorithm then  $y \stackrel{\$}{\leftarrow} A(x_1, x_2, \dots)$  means that  $y$  is the result of executing  $A$  on fresh random coins and inputs  $x_1, x_2, \dots$ . We denote by  $[A(x_1, x_2, \dots)]$  the set of all possible outputs of  $A$  on the indicated inputs, meaning the set of all strings that have a positive probability of being output by  $A$  on inputs  $x_1, x_2, \dots$ . We let  $\text{Maps}(D)$  denote the set of all functions with domain  $\{0, 1\}^*$  and range  $D$ .

DIGITAL SIGNATURE SCHEMES. We recall definitions for (standard) signature schemes [14] in the random-oracle (RO) model [12]. A signature scheme  $\mathcal{DS} = (\text{Kg}, \text{Sign}, \text{Vf})$  is specified by three algorithms, the last deterministic. Via  $(pk, sk) \stackrel{\$}{\leftarrow} \text{Kg}$ , a signer generates its public key  $pk$  and matching secret key  $sk$ , where  $H: \{0, 1\}^* \rightarrow D$  is a random oracle whose range  $D$  is a parameter of the scheme. Via  $\sigma \stackrel{\$}{\leftarrow} \text{Sign}^H(sk, m)$  the signer can generate a signature  $\sigma$  on a message  $m$ . A verifier can run  $\text{Vf}^H(pk, m, \sigma)$  to obtain a bit, with 1 indicating accept and 0 reject. The consistency (or correctness) condition is that the probability that  $\text{Vf}^H(pk, m, \sigma) = 1$  must equal one for all messages  $m \in \{0, 1\}^*$ , where the probability is over the following experiment:

$$(pk, sk) \stackrel{\$}{\leftarrow} \text{Kg}; H \stackrel{\$}{\leftarrow} \text{Maps}(D); \sigma \stackrel{\$}{\leftarrow} \text{Sign}^H(sk, m).$$

To capture security (unforgeability under chosen-message attack), we define the advantage of an adversary  $\mathcal{B}$ , denoted by  $\text{Adv}_{\mathcal{DS}}^{\text{uf-cma}}(\mathcal{B})$ , as the probability that  $\text{Vf}^H(pk, m, \sigma) = 1$ , where the probability is over the following experiment:

$$(pk, sk) \stackrel{\$}{\leftarrow} \text{Kg}; H \stackrel{\$}{\leftarrow} \text{Maps}(D); (m, \sigma) \stackrel{\$}{\leftarrow} \mathcal{B}^{\text{Sign}^H(sk, \cdot), H}.$$

To make this meaningful, we only consider adversaries that are *legitimate* in the sense that they never queried the message in their output to their sign oracle. We say that  $\mathcal{DS}$  is  $(t, q_S, q_H, \epsilon)$ -secure if no adversary  $\mathcal{B}$  running in time at most  $t$ , invoking the signature oracle at most  $q_S$  times and the random oracle at most  $q_H$  times, has advantage more than  $\epsilon$ .

## 3 Unrestricted General Aggregate Signatures

GAS SCHEMES. A general aggregate signature (GAS) scheme [1]  $\mathcal{AS} = (\text{Kg}, \text{Sign}, \text{Agg}, \text{AVf})$  consists of four algorithms, the last deterministic. The key generation and signing algorithms are exactly as for standard digital signatures. Via

$\sigma \stackrel{\$}{\leftarrow} \text{Agg}^{\text{H}}((pk_1, m_1, \sigma_1), \dots, (pk_n, m_n, \sigma_n))$ , anyone can aggregate a sequence of public key, message, and signature triples to yield an aggregate signature  $\sigma$ . A verifier can run  $\text{AVf}^{\text{H}}((pk_1, m_1), \dots, (pk_n, m_n), \sigma)$  to obtain a bit, with 1 indicating accept and 0 reject.

SECURITY. The security requirement of [1] is strong, namely that an adversary find it computationally infeasible to produce an aggregate forgery involving an honest signer, even when it can play the role of all other signers, in particular choosing their public keys, and can mount a chosen-message attack on the honest signer. To capture this, we define the advantage of an adversary  $A$  as

$$\text{Adv}_{\mathcal{AS}}^{\text{agg-uf}}(A) = \Pr \left[ \text{AVf}^{\text{H}}((pk_1, m_1), \dots, (pk_n, m_n), \sigma) = 1 \right]$$

where the probability is over the experiment

$$\begin{aligned} (pk, sk) &\stackrel{\$}{\leftarrow} \text{Kg} ; \text{H} \stackrel{\$}{\leftarrow} \text{Maps}(D) ; \\ ((pk_1, m_1), \dots, (pk_n, m_n), \sigma) &\stackrel{\$}{\leftarrow} A^{\text{Sign}^{\text{H}}(sk, \cdot), \text{H}}(pk) . \end{aligned}$$

To make this meaningful, we only consider adversaries that are *legitimate* in the sense that there must exist  $i \in \{1, \dots, n\}$  such that  $pk_i = pk$  but  $A$  never queried  $m_i$  to its signing oracle. Thus, the honest signer here is the one whose keys are  $pk, sk$ , and we are asking that the aggregate forgery include some message and signature corresponding to this honest signer, but the adversary never legitimately obtained a signature of this message. We say that  $A(t, q_S, n_{\text{max}}, q_H, \epsilon)$ -breaks  $\mathcal{AS}$  if it runs in time at most  $t$ , invokes the signature oracle at most  $q_S$  times, invokes the hash oracle at most  $q_H$  times, outputs a forgery containing at most  $n_{\text{max}}$  public-key-message pairs, and has advantage strictly greater than  $\epsilon$ . We say that  $\mathcal{AS}$  is  $(t, q_S, n_{\text{max}}, q_H, \epsilon)$ -secure if there is no adversary that  $(t, q_S, n_{\text{max}}, q_H, \epsilon)$ -breaks  $\mathcal{AS}$ .

A significant feature of this definition, highlighted in [1], is that  $A$  can choose  $pk_1, \dots, pk_n$  as it wishes, in particular as a function of  $pk$ . Unlike [10,11,9], there is no requirement that the adversary “know” the secret key corresponding to a public key it produces, and this makes the system more practical since it avoids the need for strong zero-knowledge proofs of knowledge [15] of secret keys done to the CA during key-registration. Our results continue to achieve this strong notion of security.

BILINEAR MAPS AND COCDH. Let  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$  be groups, all of the same prime order  $p$ . Let  $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  be a non-degenerate, efficiently computable bilinear map, also called a pairing. Let  $g$  be a generator of  $\mathbb{G}_2$ . For the rest of the paper, we regard  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g$  as fixed, globally known parameters, and also let  $t_{\text{exp}}$  denote the time to perform an exponentiation in  $\mathbb{G}_1$ . Note that following [3,1] we use the asymmetric setting ( $\mathbb{G}_1, \mathbb{G}_2$  are not necessarily equal) and must also assume there exists an isomorphism  $\psi: \mathbb{G}_2 \rightarrow \mathbb{G}_1$ . (The first is in order to make signatures as short as possible, and the second is required for

the security proofs.) We define the advantage of an adversary  $A$  in solving the coCDH problem as

$$\mathbf{Adv}^{\text{co-cdh}}(A) = \Pr \left[ A(g, g^a, h) = h^a : h \stackrel{\$}{\leftarrow} \mathbb{G}_1; a \stackrel{\$}{\leftarrow} \mathbb{Z}_p \right] .$$

We say that the coCDH problem is  $(t', \epsilon')$ -hard if no algorithm  $A$  running in time at most  $t'$  has advantage strictly more than  $\epsilon'$  in solving it. Note that when  $\mathbb{G}_1 = \mathbb{G}_2$ , the coCDH problem becomes the standard CDH problem in  $\mathbb{G}_1$ , whence the name.

**THE  $\mathcal{BLS}$  SCHEME.** We recall the  $\mathcal{BLS}$  standard signature scheme of [3]. The signer chooses a secret key  $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and computes the corresponding public key  $X \leftarrow g^x$ . Let  $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$  be a random oracle. The signature of message  $m$  is  $\sigma = H(m)^x$ , which can be verified by checking that  $e(\sigma, g) = e(H(m), X)$ . Regarding security, we have the following:

**Theorem 1 [3].** *If the coCDH problem is  $(t', \epsilon')$ -hard, then the  $\mathcal{BLS}$  standard signature scheme is  $(t, q_S, q_H, \epsilon)$ -secure for any  $t, q_S, q_H, \epsilon$  satisfying  $\epsilon \geq e(q_S + 1) \cdot \epsilon'$  and  $t \leq t' - t_{\text{exp}}(q_H + 2q_S)$  .* ▀

**THE GAS SCHEMES WE CONSIDER.** We consider four closely related aggregate signature schemes that we denote  $\mathcal{AS}\text{-}0, \mathcal{AS}\text{-}1, \mathcal{AS}\text{-}2, \mathcal{AS}\text{-}3$ . These schemes share common key generation and aggregation algorithms, but differ in their signing and verification algorithms. All use a random oracle  $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ . Key generation is exactly as in the  $\mathcal{BLS}$  scheme: the secret key is an exponent  $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$  and the corresponding public key is  $X = g^x$ . For  $\mathcal{AS}\text{-}0$  and  $\mathcal{AS}\text{-}1$ , the signing algorithm is the  $\mathcal{BLS}$  one, namely the signature on message  $m$  is  $\sigma = H(m)^x$ . For  $\mathcal{AS}\text{-}2$  and  $\mathcal{AS}\text{-}3$ , a signature on  $m$  under public key  $X$  is  $\sigma = H(X \| m)^x$ . For all schemes, aggregation is done by simply multiplying the signatures, i.e.  $\sigma = \prod_{i=1}^n \sigma_i$  in  $\mathbb{G}_1$ . Verification is different for each scheme. On inputs  $(X_1, m_1), \dots, (X_n, m_n), \sigma$ , the verification algorithm of  $\mathcal{AS}\text{-}0$  accepts iff  $e(\sigma, g) = \prod_{i=1}^n e(H(m_i), X_i)$ . The verification algorithms of the other schemes are depicted in Table 1. In particular,  $\mathcal{AS}\text{-}1$  rejects if  $m_1, \dots, m_n$  are not all distinct,  $\mathcal{AS}\text{-}2$  rejects if  $X_1 \| m_1, \dots, X_n \| m_n$  are not all distinct, while  $\mathcal{AS}\text{-}3$  performs no such checks.

**CONSISTENCY CONDITIONS.** The consistency condition (under what conditions correctly generated aggregates are accepted by the verifier) differs from scheme to scheme, and is in fact the place where the restrictions they make surface in a formal sense.  $\mathcal{AS}\text{-}0$  and  $\mathcal{AS}\text{-}3$  meet the natural, strongest possible consistency requirement, namely that

$$\Pr \left[ \mathbf{AVf}^H((pk_1, m_1), \dots, (pk_n, m_n), \sigma) = 1 \right] = 1$$

for all positive integers  $n$ , all messages  $m_1, \dots, m_n \in \{0, 1\}^*$  and all  $(pk_1, sk_1), \dots, (pk_n, sk_n) \in [\text{Kg}]$ , where the probability is over the experiment  $H \stackrel{\$}{\leftarrow} \text{Maps}(D); \sigma_1 \stackrel{\$}{\leftarrow} \text{Sign}^H(sk_1, m_1); \dots; \sigma_n \stackrel{\$}{\leftarrow} \text{Sign}^H(sk_n, m_n); \sigma \stackrel{\$}{\leftarrow} \text{Agg}^H((pk_1, m_1,$

$\sigma_1), \dots, (pk_n, m_n, \sigma_n)$ ). However,  $\mathcal{AS}\text{-}1$  meets this condition only when  $m_1, \dots, m_n$  are distinct and  $\mathcal{AS}\text{-}2$  when  $pk_1 || m_1, \dots, pk_n || m_n$  are distinct.

DISCUSSION OF SECURITY. An attack provided in [1] shows that  $\mathcal{AS}\text{-}0$  is insecure. In this attack, however, the forgery output by the adversary contains repeated messages. To exclude the attack, [1] defines  $\mathcal{AS}\text{-}1$ , where the aggregate verification process rejects when messages repeat. They are able to show that this suffices to guarantee security, meaning that they prove  $\mathcal{AS}\text{-}1$  is secure if the coCDH problem is hard. This is their main result. Then they suggest to alleviate the message-distinctness restriction of  $\mathcal{AS}\text{-}1$  by having each signer prepend its public key to the message before signing. However, they appear to want to argue the security of the resulting aggregate signature scheme as a corollary of their main result on the security of  $\mathcal{AS}\text{-}1$ . If so, verification still needs to check that  $X_1 || m_1, \dots, X_n || m_n$  are all distinct (otherwise, the result about  $\mathcal{AS}\text{-}1$  does not apply), leading to the  $\mathcal{AS}\text{-}2$  scheme.

As we have discussed, however, for practical reasons,  $\mathcal{AS}\text{-}3$  is a preferable scheme. But the results of [1] do not prove it secure. Here is an example that helps to see what the problem is. Suppose there was an adversary  $A$  that, on input  $pk = X$  and without making oracle query  $m$ , produced a forgery of the form  $(X, m), (X', m'), (X', m'), \sigma$ , for some  $m' \neq m$  and  $X' \neq X$ , that was accepted by the verification procedure of  $\mathcal{AS}\text{-}3$ . Since the output of  $A$  contains repeated enhanced messages, the results of [1] do not allow us to rule out the existence of  $A$ . Yet, showing that  $\mathcal{AS}\text{-}3$  meets the notion of security that we have defined does require ruling out the existence of such an  $A$ .

**Theorem 2.** *If the coCDH problem is  $(t', \epsilon')$ -hard, then the  $\mathcal{AS}\text{-}3$  aggregate signature scheme is  $(t, q_S, n_{\max}, q_H, \epsilon)$ -secure for any  $t, q_S, n_{\max}, q_H, \epsilon$  satisfying  $\epsilon \geq e(q_S + 1) \cdot \epsilon'$  and  $t \leq t' - t_{\text{exp}}(2q_H + 2q_S + 3n_{\max} + 1)$ .* ■

Our approach to the proof is different from the one used by [1] to prove that  $\mathcal{AS}\text{-}1$  is secure if coCDH is hard. They gave a direct reduction to coCDH, meaning, given an adversary attacking  $\mathcal{AS}\text{-}1$  they construct and analyze an adversary attacking coCDH. But, in so doing, they end up duplicating a lot of the proof of the security of the  $\mathcal{BLS}$  scheme as given in [3]. Instead, we reduce the security of  $\mathcal{AS}\text{-}3$  to the security of  $\mathcal{BLS}$ . That is, we prove the following:

**Lemma 3.** *If the  $\mathcal{BLS}$  standard signature scheme is  $(t', q'_S, q'_H, \epsilon')$ -secure then the  $\mathcal{AS}\text{-}3$  aggregate signature scheme is  $(t, q_S, n_{\max}, q_H, \epsilon)$ -secure for any  $t, q_S, n_{\max}, q_H, \epsilon$  satisfying  $\epsilon \geq \epsilon', q_S \leq q'_S - n_{\max}, q_H \leq q'_H$  and  $t \leq t' - t_{\text{exp}} \cdot (q_H + n_{\max} + 1)$ .* ■

Theorem 2 follows easily from Lemma 3 and Theorem 1. Our modular approach yields a simple proof even though we obtain a somewhat stronger result.

An interesting element of the proof of Lemma 3 is that it involves reducing the security of one random oracle model scheme to another one. Given a forger  $A$  against  $\mathcal{AS}\text{-}3$  that queries a random oracle, we must build a forger  $B$  against  $\mathcal{BLS}$ . But  $B$  is itself given a random oracle. The idea is that  $B$  will answer some of  $A$ 's queries via its own random oracle and directly simulate the others.

|   |   |
|---|---|
| <pre> Subroutine H-SIM(<math>M</math>) If <math>(\exists m : M = X^*    m)</math> then return <math>H_{\mathcal{BLS}}(M)</math> Else If <math>\text{HT}[M] = \perp</math> then <math>y[M] \stackrel{\\$}{\leftarrow} \mathbb{Z}_p</math>; <math>\text{HT}[M] \leftarrow \psi(g)^{y[M]}</math> Return <math>\text{HT}[M]</math>                 </pre> | <pre> Subroutine SIGN-SIM(<math>m</math>) Return <math>\text{Sign}_{\mathcal{BLS}}(x^*, X^*    m)</math>                 </pre> |
|---|---|

**Fig. 1.** The subroutines for B to simulate the random oracle  $H_{\mathcal{AS}\text{-}3}(\cdot)$  and the sign oracle  $\text{Sign}_{\mathcal{AS}\text{-}3}(x^*, \cdot)$ . Above, HT and y are associative arrays assumed initially to have value  $\perp$  everywhere.

---

*Proof (Lemma 3).* Given a forger A that  $(t, q_S, n_{\max}, q_H, \epsilon)$ -breaks  $\mathcal{AS}\text{-}3$ , consider the following forger B against the  $\mathcal{BLS}$  standard signature scheme. B is given public key  $X^* = g^{x^*}$  as input, and has access to a random oracle  $H_{\mathcal{BLS}}(\cdot)$  and a signing oracle  $\text{Sign}_{\mathcal{BLS}}(x^*, \cdot) = H_{\mathcal{BLS}}(\cdot)^{x^*}$ . It runs A on input  $X^*$  and responds to its  $H_{\mathcal{AS}\text{-}3}(\cdot)$  and  $\text{Sign}_{\mathcal{AS}\text{-}3}(x^*, \cdot)$  oracle queries using the subroutines in Fig. 1.

When A submits a query  $M$  to its random oracle  $H_{\mathcal{AS}\text{-}3}(\cdot)$ , the forger B executes H-SIM( $M$ ). We note here that in some cases the subroutine H-SIM can in turn submit queries to B’s random oracle  $H_{\mathcal{BLS}}(\cdot)$ . When A submits a query  $m$  to its sign oracle  $\text{Sign}_{\mathcal{AS}\text{-}3}(x^*, \cdot)$ , the forger B executes SIGN-SIM( $m$ ). Eventually, A halts and outputs a forgery  $(X_1, m_1), \dots, (X_n, m_n), \sigma$ . Since A is legitimate, we know that there exists  $i \in \{1, \dots, n\}$  such that  $X_i = X^*$  and  $m_i$  has never been queried to  $\text{Sign}_{\mathcal{AS}\text{-}3}(x^*, \cdot)$ . We let  $i^*$  denote the smallest integer  $i$  for which this happens. Now B defines the sets:

$$\begin{aligned}
 I &= \{ i \mid X_i = X^* \text{ and } m_i = m_{i^*} \} \\
 J &= \{ i \mid X_i = X^* \text{ and } m_i \neq m_{i^*} \} \\
 K &= \{ i \mid X_i \neq X^* \} .
 \end{aligned}$$

Note that  $I$  is non-empty since  $i^* \in I$ . Clearly, we have that  $I \cup J \cup K = \{1, \dots, n\}$  and that  $I, J, K$  are disjoint. Now, we can assume without loss of generality that  $n < p$ , because otherwise B can trivially forge and output a  $\mathcal{BLS}$  signature under  $X^*$  in time  $O(nt_{\text{exp}})$  via exhaustive search for  $x^*$ . This means that  $|I| \in \mathbb{Z}_p^*$ , and hence has an inverse modulo  $p$  that we denote by  $l$ . Now, for each  $i \in J$ , our adversary B executes SIGN-SIM( $m_i$ ) to obtain  $\sigma_i \leftarrow \text{Sign}_{\mathcal{BLS}}(x^*, X^* || m_i)$ . For each  $i \in K$ , it calls its subroutine H-SIM( $X_i || m_i$ ), thereby ensuring that  $y[X_i || m_i]$  is defined, lets  $y_i \leftarrow y[X_i || m_i]$ , and then lets  $\sigma_i \leftarrow \psi(X_i)^{y_i}$ , which we note is the  $\mathcal{BLS}$  signature of  $X_i || m_i$  under public key  $X_i$ . Finally, B lets

$$M^* \leftarrow X^* || m_{i^*} \quad \text{and} \quad \sigma^* \leftarrow \left( \sigma \cdot \prod_{i \in J \cup K} \sigma_i^{-1} \right)^l , \tag{1}$$

and outputs  $(M^*, \sigma^*)$  as its forgery.

For the analysis, we first argue that if A’s forgery is valid then B’s forgery is valid too. Assuming the former, the verification equation of  $\mathcal{AS}\text{-}3$  tells us that

$$\mathbf{e}(\sigma, g) = \prod_{i=1}^n \mathbf{e}(H_{\mathcal{AS}\text{-}3}(X_i || m_i), X_i)$$

$$\begin{aligned}
 &= \prod_{i \in I} \mathbf{e}(\mathbf{H}_{\mathcal{BLS}}(X^* \| m_{i^*}), X^*) \cdot \prod_{i \in J} \mathbf{e}(\mathbf{H}_{\mathcal{BLS}}(X^* \| m_i), X^*) \cdot \\
 &\quad \prod_{i \in K} \mathbf{e}(\mathbf{H}_{\mathcal{AS-3}}(X_i \| m_i), X_i) \\
 &= \mathbf{e}(\mathbf{H}_{\mathcal{BLS}}(X^* \| m_{i^*}), X^*)^{|I|} \cdot \prod_{i \in J \cup K} \mathbf{e}(\sigma_i, g) . \tag{2}
 \end{aligned}$$

Above, (2) is true because  $\sigma_i$ , as computed above by **B**, is the  $\mathcal{BLS}$  signature of  $X_i \| m_i$  under public key  $X_i$ , for all  $i \in J \cup K$ . We then applied the verification equation of the  $\mathcal{BLS}$  scheme. Now we see that if  $\sigma^*$  is defined by (1) then, from the above and the fact that  $|I| \cdot l \equiv 1 \pmod{p}$  we have

$$\begin{aligned}
 \mathbf{e}(\sigma^*, g) &= \mathbf{e}(\sigma, g)^l \cdot \prod_{i \in J \cup K} \mathbf{e}(\sigma_i, g)^{-l} \\
 &= \mathbf{e}(\mathbf{H}_{\mathcal{BLS}}(X^* \| m_{i^*}), X^*)^{|I| \cdot l \pmod{p}} \cdot \prod_{i \in J \cup K} \mathbf{e}(\sigma_i, g)^l \cdot \prod_{i \in J \cup K} \mathbf{e}(\sigma_i, g)^{-l} \\
 &= \mathbf{e}(\mathbf{H}_{\mathcal{BLS}}(M^*), X^*) ,
 \end{aligned}$$

which means that  $\sigma^*$  is a valid  $\mathcal{BLS}$  signature of  $M^*$  under public key  $X^*$ .

Furthermore, it is easy to see that the answers that **B** provided to the oracle queries of **A** are distributed identically to the ones that **A** would have obtained from its oracles in the game defining its advantage. The last thing we need to check is that **B** is legitimate, meaning did not query  $M^* = X^* \| m_{i^*}$  to its  $\text{Sign}_{\mathcal{BLS}}(x^*, \cdot)$  oracle. But it did not do so while answering  $\text{Sign}_{\mathcal{AS-3}}(x^*, \cdot)$  oracle queries of **A** because **A**, being legitimate itself, did not query  $m_{i^*}$  to its  $\text{Sign}_{\mathcal{AS-3}}(x^*, \cdot)$  oracle. Now **B** also called  $\text{Sign}_{\mathcal{BLS}}(x^*, \cdot)$  on  $X^* \| m_i$  for all  $i \in J$ , but by definition of  $J$ , we know that  $m_i \neq m_{i^*}$ . Putting everything together, we get  $\text{Adv}_{\mathcal{BLS}}^{\text{uf-cma}}(\mathbf{B}) \geq \text{Adv}_{\mathcal{AS-3}}^{\text{agg-uf}}(\mathbf{A})$ .

Finally, we analyze the resource usage of **B**. For  $q_S$ , we note that **B** makes sign queries in only two situations: (1) whenever **A** makes a sign query, so does **B**, and (2) once **A** outputs a forgery, **B** makes  $|J| \leq n_{\max}$  additional sign queries. For  $q_H$ , it is easy to see that **B** makes at most the same number of random oracle queries to  $\mathbf{H}_{\mathcal{BLS}}$  as **A** makes to  $\mathbf{H}_{\mathcal{AS-3}}$ . For  $t$ , notice that **B** (1) answers  $q_H$  random oracle queries, each of which results in a call to  $\mathbf{H}\text{-SIM}$ , (2) possibly makes  $n_{\max}$  more calls to  $\mathbf{H}\text{-SIM}$  after **A** outputs its forgery, and (3) computes one exponentiation in the last step to convert **A**'s forgery into its own. Thus, the claimed running time bound follows, and the proof is complete.

## Acknowledgments

Mihir Bellare was supported by NSF grants CNS-0524765, CNS-0627779, and a gift from Intel Corporation. Chanathip Namprempre was supported by the Thailand Research Fund. Gregory Neven is a Postdoctoral Fellow of the Research

Foundation Flanders, and was supported in part by the Concerted Research Action Ambiorics 2005/11 of the Flemish Government and the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. We thank the ICALP 2007 anonymous referees for their valuable comments.

## References

1. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) *Advances in Cryptology – EUROCRYPT 2003*. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
2. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
3. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
4. Intergovernmental Oceanographic Commission of UNESCO: Towards the establishment of a tsunami warning and mitigation system for the Indian Ocean (Last accessed, April 13, 2007) Available at <http://ioc3.unesco.org/indotsunami/>
5. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: A survey of two signature aggregation techniques. *RSA’s CryptoBytes* 6(2) (2003)
6. Bellare, M., Yung, M.: Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology* 9(3), 149–166 (1996)
7. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. Full version of this paper (2006) Available from <http://eprint.iacr.org/2006/285>
8. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: *ACM CCS 03*, pp. 155–164. ACM Press, New York (2003)
9. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) *EUROCRYPT 2006*. LNCS, vol. 4004, Springer, Heidelberg (2006)
10. Bellare, M., Boldyreva, A., Staddon, J.: Randomness re-use in multi-recipient encryption schemes. In: Desmedt, Y.G. (ed.) *PKC 2003*. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2002)
11. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y.G. (ed.) *PKC 2003*. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
12. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: *ACM CCS 93*, pp. 62–73. ACM Press, New York (1993)
13. Shacham, H.: *New Paradigms in Signature Schemes*. PhD thesis, Stanford University (2005)
14. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing* 17(2), 281–308 (1988)
15. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) *CRYPTO 1992*. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)

# Ring Signatures of Sub-linear Size Without Random Oracles

Nishanth Chandran, Jens Groth\*, and Amit Sahai\*\*

UCLA Computer Science Department  
4732 Boelter Hall, Los Angeles CA 90095, USA  
{nishanth, jg, sahai}@cs.ucla.edu

**Abstract.** Ring signatures, introduced by Rivest, Shamir and Tauman, enable a user to sign a message anonymously on behalf of a “ring”. A ring is a group of users, which includes the signer. We propose a ring signature scheme that has size  $\mathcal{O}(\sqrt{N})$  where  $N$  is the number of users in the ring. An additional feature of our scheme is that it has perfect anonymity.

Our ring signature like most other schemes uses the common reference string model. We offer a variation of our scheme, where the signer is guaranteed anonymity even if the common reference string is maliciously generated.

## 1 Introduction

Ring signatures, introduced by Rivest, Shamir and Tauman [RST06], enable a user to sign a message anonymously on behalf of a “ring” with the only condition that the user is a member of the ring. A ring is a collection of users chosen by the signer. The signer has to be a member of the ring but the other users do not need to cooperate and may be unaware that they are included in a ring signature.

A variety of applications have been suggested for ring signatures in previous works (see for example [RST06, Nao02, DKNS04]). The original application given was the anonymous leaking of secrets. For example, a high-ranking official in the government wishes to leak some important information to the media. The media want to verify that the source of information is valid, at the same time the official leaking it desires anonymity. Ring signatures give us a way to achieve this task, wherein the media can verify that some high-ranking government official signed the message but cannot ascertain which member actually leaked the secret. Another application is that of designated-verifier signatures [JS196]. Ring signatures enable Alice to sign an email and send it to Bob with the property that Bob cannot convince a third party that Alice actually sent him this message.

The description of the ring itself is in general linear in the number of members because it is necessary to specify the users included in the ring. Yet, one might face a situation wherein we would like to verify many different signatures on the same ring.

---

\* Supported by NSF ITR/Cybertrust grant No. 0456717.

\*\* Supported by grant No. 0456717 from the NSF ITR and Cybertrust programs, an equipment grant from Intel, and an Alfred P. Sloan Foundation Research Fellowship.



In this case, the size of the ring-signature being sub-linear is quite useful.<sup>1</sup> Most ring signature schemes known today are of linear size in the number of ring members, the only exception being the scheme in [DKNS04], which is independent of the size of the ring. Apart from [CWLY06, BKM06, SW06, Boy07], to the best of our knowledge, all other constructions (including [DKNS04]) are in the random oracle model. The scheme in [CWLY06] is based on a strong new assumption, while in [BKM06], the scheme uses generic ZAPs for NP, thus making it impractical. Shacham and Waters [SW06] give a construction of linear size that is secure under the computational setting of the definitions in [BKM06]. Boyen [Boy07] gives a linear size ring signature in the common random string model with perfect anonymity. Our goal is to construct a sub-linear size ring-signature scheme with perfect anonymity without random oracles.

## 1.1 Our Contribution

We give the first ring signature of sub-linear size without random oracles. Our scheme is based on composite order groups with a bilinear map. Security is based on the strong Diffie-Hellman assumption [BB04] and the subgroup decision assumption [BGN05]. Our scheme has perfect anonymity in the common reference string model. To reduce the amount of trust in the common reference string, we also offer a variant of our scheme that gives an unconditional guarantee of anonymity even if the common reference string is generated maliciously. Both schemes have ring signatures of size  $\mathcal{O}(k\sqrt{N})$  bits, where  $N$  is the number of users in the ring and  $k$  is a security parameter.

TECHNIQUE. The broad idea behind the scheme is as follows: Let the number of members in the ring be  $N$ . To compute a ring signature, the signer first chooses a random one-time signature key and issues a signature on the message using this one-time signing key. Both the public key of the one-time signature and the signature are published. Next, the signer validates the one-time signature key. In other words, she signs the one-time signature key with her own signing key. This validation signature has to be hidden for anonymity. The signer, hence makes two perfectly hiding commitments to her verification key and the validation signature and publishes these values. She then makes non-interactive witness-indistinguishable (NIWI) proofs using techniques from [GOS06, BW06, GS06] that the commitments indeed contain a verification key and a signature on the one-time signature verification key respectively. Finally, the signer will prove that the committed verification key belongs to the ring. The main novelty in our scheme is a sub-linear size proof for a commitment containing one out of  $N$  verification keys. This proof relies on a technique akin to one-round private information retrieval (PIR) with  $\mathcal{O}(\sqrt{N})$  communication complexity, which is used to get a commitment to the verification key.

## 2 Ring Signatures – Definitions

[BKM06] contains a comprehensive classification of ring signature definitions. We achieve security under the strongest of these definitions. In the following, we will

<sup>1</sup> Generally speaking, sub-linear size ring signatures are useful when we can amortize the cost of describing the ring itself over many signatures.

modify their definitions in order to include a common reference string and to define information theoretical anonymity.

**Definition 1 (Ring signature).** A ring signature scheme consists of a quadruple of PPT algorithms (CRSGen, Gen, Sign, Verify) that respectively, generate the common reference string, generate keys for a user, sign a message, and verify the signature of a message.

- $\text{CRSGen}(1^k)$ , where  $k$  is a security parameter, outputs the common reference string  $\rho$ .
- $\text{Gen}(\rho)$  is run by the user. It outputs a public verification key  $vk$  and a private signing key  $sk$ .
- $\text{Sign}_{\rho,sk}(M, R)$  outputs a signature  $\sigma$  on the message  $M$  with respect to the ring  $R = (vk_1, \dots, vk_N)$ . We require that  $(vk, sk)$  is a valid key-pair output by  $\text{Gen}$  and that  $vk \in R$ .
- $\text{Verify}_{\rho,R}(M, \sigma)$  verifies a purported signature  $\sigma$  on a message  $M$  with respect to the ring of public keys  $R$ .

The quadruple  $(\text{CRSGen}, \text{Gen}, \text{Sign}, \text{Verify})$  is a ring signature with perfect anonymity if it has perfect correctness, computational unforgeability and perfect anonymity as defined below.

**Definition 2 (Perfect correctness).** We require that a user can sign any message on behalf of a ring where she is a member. A ring signature  $(\text{CRSGen}, \text{Gen}, \text{Sign}, \text{Verify})$  has perfect correctness if for all adversaries  $\mathcal{A}$  we have:

$$\Pr \left[ \rho \leftarrow \text{CRSGen}(1^k); (vk, sk) \leftarrow \text{Gen}(\rho); (M, R) \leftarrow \mathcal{A}(\rho, vk, sk); \right. \\ \left. \sigma \leftarrow \text{Sign}_{sk}(M, R) : \text{Verify}_{\rho,R}(M, \sigma) = 1 \vee vk \notin R \right] = 1.$$

**Definition 3 (Unforgeability).** A ring signature scheme  $(\text{CRSGen}, \text{Gen}, \text{Sign}, \text{Verify})$  is unforgeable (with respect to insider corruption) if it is infeasible to forge a ring signature on a message without controlling one of the members in the ring. Formally, it is unforgeable when there is a negligible function  $\epsilon$  so for any non-uniform polynomial time adversaries  $\mathcal{A}$  we have:

$$\Pr \left[ \rho \leftarrow \text{CRSGen}(1^k); (M, R, \sigma) \leftarrow \mathcal{A}^{\text{VKGen}, \text{Sign}, \text{Corrupt}}(\rho) : \right. \\ \left. \text{Verify}_{\rho,R}(M, \sigma) = 1 \right] < \epsilon(k),$$

- $\text{VKGen}$  on query number  $i$  selects a randomizer  $w_i$ , runs  $(vk_i, sk_i) \leftarrow \text{Gen}(\rho; w_i)$  and returns  $vk_i$ .
- $\text{Sign}(\alpha, M, R)$  returns  $\sigma \leftarrow \text{Sign}_{\rho,sk_\alpha}(M, R)$ , provided  $(vk_\alpha, sk_\alpha)$  has been generated by  $\text{VKGen}$  and  $vk_\alpha \in R$ .
- $\text{Corrupt}(i)$  returns  $w_i$  (from which  $sk_i$  can be computed) provided  $(vk_i, sk_i)$  has been generated by  $\text{VKGen}$ .
- $\mathcal{A}$  outputs  $(M, R, \sigma)$  such that  $\text{Sign}$  has not been queried with  $(*, M, R)$  and  $R$  only contains keys  $vk_i$  generated by  $\text{VKGen}$  where  $i$  has not been corrupted.

**Definition 4 (Perfect anonymity).** A ring signature scheme  $(\text{CRSGen}, \text{Gen}, \text{Sign}, \text{Verify})$  has perfect anonymity, if a signature on a message  $M$  under a ring  $R$  and key  $vk_{i_0}$  looks exactly the same as a signature on the message  $M$  under the ring  $R$  and key  $vk_{i_1}$ . This means that the signer's key is hidden among all the honestly generated keys in the ring. Formally, we require that for any adversary  $\mathcal{A}$ :

$$\begin{aligned} & \Pr \left[ \rho \leftarrow \text{CRSGen}(1^k); (M, i_0, i_1, R) \leftarrow \mathcal{A}^{\text{Gen}(\rho)}(\rho); \right. \\ & \quad \left. \sigma \leftarrow \text{Sign}_{\rho, sk_{i_0}}(M, R) : \mathcal{A}(\sigma) = 1 \right] \\ = & \Pr \left[ \rho \leftarrow \text{CRSGen}(1^k); (M, i_0, i_1, R) \leftarrow \mathcal{A}^{\text{Gen}(\rho)}(\rho); \right. \\ & \quad \left. \sigma \leftarrow \text{Sign}_{\rho, sk_{i_1}}(M, R) : \mathcal{A}(\sigma) = 1 \right], \end{aligned}$$

where  $\mathcal{A}$  chooses  $i_0, i_1$  such that  $(vk_{i_0}, sk_{i_0}), (vk_{i_1}, sk_{i_1})$  have been generated by the oracle  $\text{Gen}(\rho)$ .

We remark that perfect anonymity implies anonymity against full key exposure which is the strongest definition of anonymity in [BKM06].

### 3 Preliminaries

We make use of bilinear groups of composite order. These were introduced by Boneh, Goh and Nissim [BGN05] and can for instance be based on elliptic curves and the modified Weil-pairing from Boneh and Franklin [BF03]. Let  $\text{Gen}_{\text{BGN}}$  be a randomized algorithm that outputs  $(p, q, G, G_T, e, g)$  so we have:

- $G$  is a multiplicative cyclic group of order  $n := pq$
- $g$  is a generator of  $G$
- $G_T$  is a multiplicative group of order  $n$
- $e : G \times G \rightarrow G_T$  is an efficiently computable map with the following properties:
  - Bilinear:  $\forall u, v \in G$  and  $a, b \in \mathbb{Z}_n : e(u^a, v^b) = e(u, v)^{ab}$
  - Non-degenerate:  $e(g, g)$  is a generator of  $G_T$  whenever  $g$  is a generator of  $G$
- The group operations on  $G$  and  $G_T$  can be performed efficiently

We will write  $G_p$  and  $G_q$  for the unique subgroups of  $G$  that have respectively order  $p$  and order  $q$ . Observe,  $u \mapsto u^q$  maps  $u$  into the subgroup  $G_p$ .

We base our ring signature scheme on two assumptions - namely, the strong Diffie-Hellman Assumption [BB04] in  $G_p$  and the subgroup decision assumption [BGN05].

**SUBGROUP DECISION ASSUMPTION.** Informally, in the above setting of composite order groups, the subgroup decision assumption holds if random elements from  $G$  and  $G_q$  are computationally indistinguishable. Formally, for generator  $\text{Gen}_{\text{BGN}}$ , the subgroup decision assumption holds if there is a negligible function  $\epsilon$  so for any non-uniform polynomial time adversary  $\mathcal{A}$ :

$$\Pr \left[ (p, q, G, G_T, e, g) \leftarrow \text{Gen}_{\text{BGN}}(1^k); n := pq; r \leftarrow \mathbb{Z}_n^*; h := g^r : \right.$$

$$\begin{aligned}
 & \mathcal{A}(n, G, G_T, e, g, h) = 1 \Big] \\
 - \Pr & \left[ (p, q, G, G_T, e, g) \leftarrow \text{Gen}_{\text{BGN}}(1^k); n := pq; r \leftarrow \mathbb{Z}_q^*; h := g^{pr} : \right. \\
 & \left. \mathcal{A}(n, G, G_T, e, g, h) = 1 \right] \leq \epsilon(k).
 \end{aligned}$$

**STRONG DIFFIE-HELLMAN ASSUMPTION IN  $G_p$ .** The strong Diffie-Hellman assumption holds in  $G_p$  if there is a negligible function  $\epsilon$  so for all non-uniform adversaries that run in polynomial time in the security parameter:

$$\begin{aligned}
 \Pr & \left[ (p, q, G, G_T, e, g) \leftarrow \text{Gen}_{\text{BGN}}(1^k); x \leftarrow \mathbb{Z}_p^* : \right. \\
 & \left. \mathcal{A}(p, q, G, G_T, e, g^q, g^{qx}, g^{qx^2}, \dots) = (c, g^{\frac{q}{x+c}}) \in \mathbb{Z}_p \times G_p \right] < \epsilon(k).
 \end{aligned}$$

**UNDERLYING SIGNATURE SCHEME.** Boneh and Boyen [BB04] suggest two signature schemes. One that is secure against weak chosen message attack, see below, and one which is secure against adaptive chosen message attack. We will use the scheme that is secure against weak chosen message attack, since it has a shorter public key and this leads to a simpler and more efficient ring signature.

We define the scheme to be secure against weak message attack if there is a negligible function  $\epsilon$  so for all non-uniform polynomial time interactive adversaries  $\mathcal{A}$ :

$$\begin{aligned}
 \Pr & \left[ (M_1, \dots, M_q) \leftarrow \mathcal{A}(1^k); (vk, sk) \leftarrow \text{KeyGen}(1^k); \sigma_i \leftarrow \text{Sign}_{sk}(M_i); \right. \\
 & \left. (M, \sigma) \leftarrow \mathcal{A}(vk, \sigma, \dots, \sigma_q) : \text{Verify}_{vk}(M, \sigma) = 1 \text{ and } M \notin \{M_1, \dots, M_q\} \right] < \epsilon(k).
 \end{aligned}$$

The Boneh-Boyen signature scheme adapted to the composite order bilinear group model is weak message attack secure under the strong Diffie-Hellman assumption.

- **Key generation:** Given a group  $(p, q, G, G_T, e, g)$  we pick a random  $sk \leftarrow \mathbb{Z}_n^*$  and compute  $vk := g^{sk}$ . The key pair is  $(vk, sk)$ .
- **Signing:** Given a secret key  $sk \in \mathbb{Z}_n^*$  and a message  $M \in \{0, 1\}^\ell$ , output the signature  $\sigma := g^{\frac{1}{sk+M}}$ . By convention,  $1/0$  is defined to be 0 so that in the unlikely event that  $sk + M = 0$ , we have  $\sigma := 1$ . We require  $\ell < |p|$ , this is quite reasonable since we can always use a cryptographic hash-function to shorten the message we sign.
- **Verification:** Given a public key  $vk$ , a message  $M \in \{0, 1\}^\ell$  and a signature  $\sigma \in G$ , verify that

$$e(\sigma, vk \cdot g^M) = e(g, g).$$

If equality holds output “Accept”. Otherwise, output “Reject”.

Boneh and Boyen [BB04] prove that their signature scheme is existentially unforgeable under weak chosen message attack provided the strong Diffie-Hellman assumption holds in prime order groups. This proof translates directly to the composite group order model. Our concern is only whether a signature is forged in the order  $p$  subgroup  $G_p$ , i.e., an adversary that knows  $p$  and  $q$  finds  $(M, \sigma)$  so  $e(vkg^M, \sigma)^q = e(g, g)^q$ . As in

[\[BB04\]](#) it can be shown to be infeasible to forge a signature in  $G_p$  under a weak chosen message attack assuming the strong Diffie-Hellman assumption holds in  $G_p$ .

**A COMMITMENT/ENCRYPTION SCHEME.** We use a commitment/encryption scheme based on the subgroup decision assumption from [\[BGN05\]](#). The public key will be a description of the composite order group as well as an element  $h$ . The element  $h$  is a random element, chosen to have order  $n$  (perfect hiding commitment) or order  $q$  (encryption). The subgroup decision assumption implies that perfect hiding commitment keys and encryption keys are indistinguishable.

To commit to a message  $m \in G$ , we pick  $r \leftarrow \mathbb{Z}_n$  at random and compute the commitment  $c := mh^r$ . When  $h$  has order  $n$ , this is a perfectly hiding commitment to  $m$ . However, if  $h$  has order  $q$ , the commitment uniquely determines  $m$ 's projection on  $G_p$ . Let  $\lambda$  be chosen so  $\lambda = 1 \pmod p$  and  $\lambda = 0 \pmod q$ . Given the factorization of  $n$ , we can compute

$$m_p = c^\lambda = m^\lambda h^{\lambda r} = m^\lambda.$$

We can also commit to a message  $m \in \mathbb{Z}_n$  by computing  $g^m h^r$ . If  $h$  has order  $n$ , then this is a perfectly hiding Pedersen commitment. If  $h$  has order  $q$ , then the commitment uniquely determines  $m \pmod p$ .

**NON-INTERACTIVE WITNESS-INDISTINGUISHABLE PROOFS.** A non-interactive proof enables us to prove that a statement is true. The proof should be complete, meaning that if we know a witness for the statement being true, then we can construct a proof. The proof should be sound, meaning that it is impossible to construct a proof for a false statement. We will use non-interactive proofs that have perfect witness-indistinguishability. This means that given two different witnesses for the statement being true, the proof reveals no information about which witness we used when we constructed the proof.

We will use the public key for the perfectly hiding commitment scheme described above as a common reference string for our NIWI proofs. When  $h$  has order  $n$  we get perfect witness-indistinguishability. However, if  $h$  has order  $q$ , then the proof has perfect soundness in  $G_p$ .

One type of statement that we will need to prove is that a commitment  $c$  is of the form  $c = g^m h^r$  for  $m \in \{0, 1\}$ . Boyen and Waters [\[BW06\]](#), building on [\[GOS06\]](#), give a non-interactive witness-indistinguishable proof for this kind of statement,  $\pi = (g^{2m-1} h^r)^r$ , which is verified by checking  $e(c, cg^{-1}) = e(h, \pi)$ . When  $h$  has order  $n$ , this proof has perfect witness-indistinguishability, because  $\pi$  is uniquely determined from the verification equation so all witnesses must give the same proof. On the other hand, if  $h$  has order  $q$ , then the verification shows that  $e(c, cg^{-1})$  has order  $q$ . This implies  $m = 0 \pmod p$  or  $m = 1 \pmod p$ .

We will also need non-interactive witness-indistinguishable proofs for more advanced statements. Groth and Sahai [\[GS06\]](#) show that there exist very small non-interactive witness-indistinguishable proofs for a wide range of statements. These proofs have perfect completeness on both types of public key for the commitment scheme, perfect soundness in  $G_p$ , when  $h$  has order  $q$ , and perfect witness-indistinguishability when  $h$  has order  $n$ .

## 4 Sub-linear Size Ring Signature Scheme Construction

We will give a high level description of the ring signature. We have a signer that knows  $sk_\alpha$  corresponding to one of the verification keys in the ring  $R = \{vk_1, \dots, vk_N\}$  and wants to sign a message  $M$ . The verification keys are for the Boneh-Boyen signature scheme. There are three steps in creating a signature:

1. The signer picks one-time signature keys,  $(otvk, otsk) \leftarrow \text{KeyGen}_{\text{one-time}}(1^k)$ . The message  $M$  will be signed with the one-time signature scheme. The verification key  $otvk$  and the one-time signature will both be public. The signer will certify  $otvk$  by signing it with a Boneh-Boyen signature under  $vk_\alpha$ .
2. The signer needs to hide  $vk_\alpha$  and the certifying signature on  $otvk$ . She will therefore make two perfectly hiding commitments to respectively  $vk_\alpha$  and the signature. Using techniques from [GS06] she makes a non-interactive witness-indistinguishable proof that the commitments contain a verification key and a signature on  $otvk$ .
3. Finally, the signer will prove that the committed verification key belongs to the ring. The main novelty in our scheme is this sub-linear size proof. She arranges  $R$  in an  $\nu \times \nu$  matrix, where  $\nu = \sqrt{N}$ . She commits to the row of the matrix that contains  $vk_\alpha$  and makes a non-interactive witness-indistinguishable proof for having done this. She then makes a non-interactive witness-indistinguishable proof that the committed verification key appears in this row.

We now present a detailed description of the ring signature scheme. CRSGen generates a common reference that contains the description of a composite order group and a public key for the perfectly hiding commitment scheme.

### CRSGen( $1^k$ )

1.  $(p, q, G, G_T, e, g) \leftarrow \text{Gen}_{\text{BGN}}(1^k)$
2.  $n := pq; x \leftarrow \mathbb{Z}_n^*; h := g^x$
3. Output  $(n, G, G_T, e, g, h)$  /\* Perfectly hiding commitment scheme

The users' key generation algorithm Gen takes as input a common reference string and outputs a signing public-private key pair  $(vk, sk)$ . In our case, it will output keys for the Boneh-Boyen signature scheme that is secure against weak message attack.

### Gen( $n, G, G_T, e, g, h$ )

1.  $sk \leftarrow \mathbb{Z}_n^*; vk := g^{sk}$
2. Output  $(vk, sk)$  /\* Boneh-Boyen signature scheme with public key  $(g, vk)$

A user with keys  $(vk_\alpha, sk_\alpha)$  wants to sign a message  $M$  under the ring  $R = \{vk_1, \dots, vk_N\}$  of size  $N$ . Let  $i, j$  be values such that  $\alpha = (i - 1)\nu + j$ , where  $\nu = \sqrt{N}$ .<sup>2</sup> It is useful to think of  $R$  as a  $\nu \times \nu$  matrix. Then  $vk_\alpha = vk_{(i-1)\nu+j}$  is the entry in row  $i$  and column  $j$ .

<sup>2</sup> Without loss of generality we assume  $N$  is a square. If  $N$  is not a square, we can simply copy  $vk_1$  sufficiently many times to make  $N$  a square.

$\text{Sign}_{(n,G,G_T,e,g,h,sk_\alpha)}(M,R)$

1.  $(otvk, ot sk) \leftarrow \text{KeyGen}_{\text{one-time}}(1^k)$ ;  $\sigma_{\text{one-time}} \leftarrow \text{Sign}_{ot sk}(M,R)$

*/\** This was step 1 in the high level description: a one-time signature on the message and the ring. The pair  $(otvk, \sigma_{\text{one-time}})$  will be public.

2.  $r \leftarrow \mathbb{Z}_n$ ;  $C := vk_\alpha h^r$ ;  $\sigma_\alpha := g^{\frac{1}{sk_\alpha + otvk}}$ ;  $s \leftarrow \mathbb{Z}_n$ ;  $L := \sigma_\alpha h^s$ ;  $\pi_L := g^{\frac{r}{sk_\alpha + otvk} + (sk_\alpha + otvk)s} \cdot h^{rs}$

*/\** This was step 2 in the high level description.  $\sigma_\alpha$  is the signer's certifying signature on  $otvk$ .  $C, L$  are perfectly hiding commitments to respectively  $vk_\alpha$  and  $\sigma_\alpha$ .  $\pi_L$  is a NIWI proof [GS06] that  $C, L$  contain respectively a verification key and a signature on  $otvk$ . All that remains is to make a NIWI proof that  $C$  contains some  $vk_\alpha \in R$  without revealing which one. The rest of the protocol is this NIWI proof.

3.  $r_l \leftarrow \mathbb{Z}_n$ ;  $C_l := h^{r_l}$ ;  $\pi_l^C := (g^{-1} h^{r_l})^{r_l}$  for  $0 \leq l < \nu$ ,  $l \neq i-1$ ;  
 $r_{i-1} := -\sum_{l \neq i-1} r_l$ ;  $C_{i-1} := g h^{r_{i-1}}$ ;  $\pi_{i-1}^C := (g h^{r_{i-1}})^{r_{i-1}}$

*/\** The commitments  $C_0, \dots, C_{\nu-1}$  are chosen so  $C_{i-1}$  is a commitment to  $g$ , whereas the others are commitments to 1. The proofs  $\pi_0, \dots, \pi_{\nu-1}$  are NIWI proofs [GOS06, BW06] that each  $C_0, \dots, C_{\nu-1}$  contains either 1 or  $g$ . Since the commitments have been chosen such that  $\prod_{l=0}^{\nu-1} C_l = g$ , this tells the verifier that there is exactly one  $C_{i-1}$  that contains  $g$ , while the other commitments contain 1. We will use this in a PIR-like fashion to pick out row  $i$  in the  $\nu \times \nu$  matrix  $R$ . Observe, for all  $1 \leq m \leq \nu$  we have  $A_m := \prod_{l=0}^{\nu-1} e(C_l, vk_{l\nu+m}) = e(g, vk_{(i-1)\nu+m}) e(h, \prod_{l=0}^{\nu-1} vk_{l\nu+m}^{r_l})$ , which is a commitment to  $e(g, vk_{(i-1)\nu+m})$ .

4.  $s_m \leftarrow \mathbb{Z}_n$ ;  $B_m := vk_{\nu(i-1)+m} h^{s_m}$ ;  $\pi_m^B := g^{-s_m} \cdot \prod_{l=0}^{\nu-1} vk_{l\nu+m}^{r_l}$  for  $1 \leq m \leq \nu$

*/\**  $B_1, \dots, B_\nu$  are commitments to the verification keys in row  $i$  of  $R$ . Recall  $A_1, \dots,$

$A_\nu$  contain row  $i$  of  $R$  paired with  $g$ .  $\pi_1^B, \dots, \pi_\nu^B$  are NIWI proofs [GS06] that  $B_1, \dots, B_\nu$  contain elements that paired with  $g$  give the contents of  $A_1, \dots, A_\nu$ . This demonstrates to the verifier that  $B_1, \dots, B_\nu$  indeed does contain row  $i$  of  $R$ .

5.  $t_m \leftarrow \mathbb{Z}_n$ ;  $D_m := h^{t_m}$ ;  $\pi_m^D := (g^{-1} h^{t_m})^{t_m}$  for  $1 \leq m \leq \nu$ ,  $m \neq j$   
 $t_j := -\sum_{m \neq j} t_m$ ;  $D_j := g h^{t_j}$ ,  $\pi_j^D := (g h^{t_j})^{t_j}$

*/\**  $D_1, \dots, D_\nu$  are commitments so  $D_j$  contains  $g$ , and the other commitments contain 1. The NIWI proofs [GOS06, BW06]  $\pi_1^D, \dots, \pi_\nu^D$  convince the verifier that  $D_1, \dots, D_\nu$  contain 1 or  $g$ . Combining this with  $\prod_{m=1}^{\nu} D_m = g$  shows that exactly one  $D_j$  is a commitment to  $g$ , while the others contain 1.

6.  $\pi_C := g^{s_j - r} \prod_{m=1}^{\nu} vk_{(i-1)\nu+m}^{t_m} h^{s_m t_m}$

$/* A := \prod_{m=1}^{\nu} e(B_m, D_m) = e(g, vk_{(i-1)\nu+j})e(h, g^{s_j} \prod_{m=1}^{\nu} vk_{(i-1)\nu+m}^{t_m} h^{s_m t_m})$   
 is a commitment to  $e(g, vk_{\alpha})$ .  $\pi_C$  is a NIWI proof [GOS06] that the content of  $C$  paired with  $g$  corresponds to the content in  $A$ .

7. Output the signature  $\sigma := \left( otvk, \sigma_{\text{one-time}}, C, L, \pi_L, \{C_0, \dots, C_{\nu-1}\}, \{\pi_0^C, \dots, \pi_{\nu-1}^C\}, \{B_1, \dots, B_{\nu}\}, \{\pi_1^B, \dots, \pi_{\nu}^B\}, \{D_1, \dots, D_{\nu}\}, \{\pi_1^D, \dots, \pi_{\nu}^D\}, \pi_C \right)$ .

Verify $_{(n,G,G_T,e,g,h,R)}(M, \sigma)$

1. Verify that  $\sigma_{\text{one-time}}$  is a one-time signature of  $M, R$  under  $otvk$ .
2. Verify that  $e(L, Cg^{otvk}) = e(g, g)e(h, \pi_L)$ .
3. Verify that  $e(C_l, C_l g^{-1}) = e(h, \pi_l^C)$  for all  $0 \leq l < \nu$  and  $\prod_{l=1}^{\nu} C_l = g$ .
4. Compute  $A_m := \prod_{l=1}^{\nu} e(C_l, vk_{(l-1)\nu+m})$  and verify  $A_m = e(g, B_m)e(h, \pi_m^B)$  for all  $1 \leq m \leq \nu$ .
5. Verify that  $e(D_m, D_m g^{-1}) = e(h, \pi_m^D)$  for all  $1 \leq m \leq \nu$  and  $\prod_{m=1}^{\nu} D_m = g$ .
6. Compute  $A := \prod_{m=1}^{\nu} e(B_m, D_m)$  and verify  $A = e(C, g)e(h, \pi_C)$ .
7. ‘‘Accept’’ if all the above steps verify correctly, otherwise ‘‘Reject’’.

**Theorem 1.** *The scheme presented in the previous section is a ring signature scheme with perfect correctness, perfect anonymity and computational unforgeability under the subgroup decision assumption, the strong Diffie-Hellman assumption and the assumption that the one-time signature is unforgeable.*

*Sketch of Proof.* Perfect correctness follows by inspection. Perfect anonymity follows from the fact that  $otvk$  and  $\sigma_{\text{one-time}}$  are generated the same way, no matter which signing key we use, and the fact that when  $h$  has order  $n$ , then all the commitments are perfectly hiding and the proofs are perfectly witness-indistinguishable [GOS06, BW06, GS06].

Computational unforgeability can be proven in three steps. By the subgroup decision assumption it is possible to switch from using  $h$  of order  $n$  in the common reference string to use  $h$  of order  $q$  with only negligible change in the probability of a forgery happening. The commitments are now perfectly binding in  $G_p$  and the NIWI proofs are perfectly sound in  $G_p$  [GOS06, BW06, GS06], so  $C$  contains some uncorrupt  $vk_{\alpha} \in R$  and  $L$  contains a signature  $\sigma_{\alpha}$  on  $otvk$  under  $vk_{\alpha}$ . By the properties of the one-time signature scheme,  $otvk$  has not been used in any other signature, and thus  $\sigma_{\alpha}$  is a forged Boneh-Boyen signature on  $otvk$ . By the strong Diffie-Hellman assumption this probability is negligible.  $\square$

## 5 Untrusted Common Reference String Model

Suppose we do not trust the common reference string. There are two possible problems: maybe it is possible to forge signatures, or maybe the ring signatures are not anonymous. The possibility of forgery can in many cases be viewed as an extended ring signature, we know that one of the  $N$  ring-members or the key generator signed the message. This may not be so problematic, if for instance one of the ring members was the key generator this is not a problem since that member can sign anyway. A breach of anonymity seems more problematic. If we consider the example from the introduction,



where a high-ranking official wants to leak a secret to the media, she needs to have strong guarantees of her anonymity. We will modify our scheme to get a (heuristically) unconditional guarantee of anonymity.

In the scheme presented earlier the common reference string is  $\rho = (n, G, G_T, e, g, h)$ . If we generate the groups as described in [BGN05] it is easy to verify that we have a group of order  $n$  with a bilinear map  $e$ , where all group operations can be computed efficiently. It is also easy to find a way to represent the group elements, so we can check that  $g, h \in G$  [GOS06]. What is hard to check is how many prime factors  $n$  has and what the order of  $g$  and  $h$  is. We make the following observation, which follows from the proof of anonymity: If  $h$  has order  $n$ , then the ring signature has perfect anonymity. We will therefore not include  $h$  in the common reference string but instead provide a method for the signer to choose a full order  $h$  as she creates the ring signature.

To get anonymity,  $h$  should have order  $n$ . If we pick a random element in  $G$  there is overwhelming probability that it has order  $n$ , unless  $n$  has a small prime factor. Lenstra’s ECM factorization algorithm [Len87] heuristically takes  $O(e^{(1+o(1))\sqrt{(\ln p)(\ln \ln p)}})$  steps to find the prime factor  $p$ . Therefore, it is heuristically possible to verify that  $n$  only has superpolynomial prime factors and we can pick random elements that with overwhelming probability have order  $n$ .

We will modify the key generation such that a user also picks a random element  $h_i \in G$  when creating her key. The signer’s anonymity will be guaranteed if the element she picks has order  $n$ . When she wants to issue a signature, she picks  $t \leftarrow \mathbb{Z}_n$  at random and uses  $h := \prod_{i=1}^N h_i^{t^{i-1}}$ . We will argue in the proof of Theorem 2 that with overwhelming probability over the choice of  $t$ , that element  $h$  she generates this way has order  $n$ . Using this  $h$  she then creates the ring signature as described in the previous section.

### 5.1 Ring Signature with Unconditional Anonymity

Our modified ring signature scheme  $(\text{CRSGen}', \text{Gen}', \text{Sign}', \text{Verify}')$  works as follows:

- $\text{CRSGen}'(1^k)$  outputs  $\rho' := (n, G, G_T, e, g, h') \leftarrow \text{CRSGen}(1^k)$
- $\text{Gen}'(\rho')$  uses Lenstra’s ECM factorization algorithm to check that  $n$  has no polynomial size prime factors. It runs  $(vk_i, sk_i) \leftarrow \text{Gen}(\rho')$  and picks  $h_i$  at random from  $G$ . It sets  $vk'_i := (vk_i, h_i)$  and outputs  $(vk'_i, sk_i)$ <sup>3</sup>
- $\text{Sign}'_{\rho', sk_\alpha}(M, R')$  sets  $R := (vk_1, \dots, vk_N)$  for  $R' = ((vk_1, h_1), \dots, (vk_N, h_N))$ . It picks  $t \leftarrow \mathbb{Z}_n$  and sets  $h := \prod_{i=1}^N h_i^{t^{i-1}}$ . It sets  $\rho := (n, G, G_T, e, g, h)$  and creates a ring signature  $\sigma \leftarrow \text{Sign}_{\rho, sk_\alpha}(M, R)$ . It outputs  $\sigma' := (t, \sigma)$ .
- $\text{Verify}'_{\rho', R'}(M, \sigma')$  sets  $R := (vk_1, \dots, vk_N)$  and  $h := \prod_{i=1}^N h_i^{t^{i-1}}$  as the signing algorithm. It sets  $\rho := (n, G, G_T, e, g, h)$  and outputs the response of  $\text{Verify}_{\rho, R}(M, \sigma)$ .

<sup>3</sup> For practical purposes, say with 1024-bit  $n$  and ring-size less than 10000, checking that  $n$  has no prime factors smaller than 40 bits is sufficient to guarantee that each time the user signs a message there is less than one in a million risk of the signature not being perfectly anonymous. Since Lenstra’s ECM factorization algorithm is only run once during key generation and is reasonably efficient when looking for 40-bit prime factors this cost is reasonable.

**Theorem 2.** *The quadruple  $(\text{CRSGen}', \text{Gen}', \text{Sign}', \text{Verify}')$  is a ring signature scheme with perfect correctness, heuristic statistical anonymity and computational unforgeability under the subgroup decision and strong Diffie-Hellman assumptions.*

*Sketch of proof.* To prove computational unforgeability we will modify  $\text{Gen}'$  such that it picks  $h_i$  of order  $q$ . Using the groups suggested in [BGN05] we can construct convincing randomness that would lead  $\text{Gen}'$  to pick such an  $h_i$ . We can therefore answer any corruption queries the adversary makes. By the subgroup decision assumption, no non-uniform polynomial time adversary can distinguish between seeing correctly generated  $h_i$ 's of order  $n$  and  $h_i$ 's of order  $q$ . It must therefore have at most negligibly smaller chance of producing a forgery after our modification. Now  $h = \prod_{i=1}^N h_i^{t^{i-1}}$  has order  $q$  for any  $t \in \mathbb{Z}_n$ . The proof of Theorem 1 shows that a polynomial time adversary with has negligible chance of producing a forgery on  $h$  of order  $q$ .

We will now prove heuristic statistical anonymity, even when the common reference string is maliciously generated by the adversary. Consider an honest signer with keys  $(vk_\alpha, h_\alpha), sk_\alpha$ . From the run of Lenstra's ECM factorization algorithm we know heuristically that  $n$  has no polynomial size prime factors. Therefore, with overwhelming probability the randomly chosen  $h_\alpha$  has order  $n$ . We will argue that with overwhelming probability over the choice of  $t$ , the signer picks  $h$  that has order  $n$ . When  $h$  has order  $n$  all commitments will be perfectly hiding and all proofs will be perfectly witness-indistinguishable [GS06], so we will get perfect anonymity.

It remains to argue that with overwhelming probability over  $t$  the element  $h = \prod_{i=1}^N h_i^{t^{i-1}}$  has order  $n$ . Consider a generator  $\gamma$  for  $G$  and let  $x_1, \dots, x_N$  be the discrete logarithms of  $h_1, \dots, h_N$  with respect to  $\gamma$ . We wish to argue that for any prime factor  $p|n$  we have  $\sum_{i=1}^N t^{i-1}x_i \neq 0 \pmod p$ .

Given a prime  $p|n$  we will show that there is at most  $N - 1$  choices of  $t \pmod p$  so  $\sum_{i=1}^N t^{i-1}x_i = 0 \pmod p$ . To see this, consider the following system of linear equations:

$$V\mathbf{x} = \begin{pmatrix} 1 & t_1 & t_1^2 & \dots & t_1^{N-1} \\ 1 & t_2 & t_2^2 & \dots & t_2^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_N & t_N^2 & \dots & t_N^{N-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

$V$  is a Vandermonde matrix and has non-zero determinant if all  $t_1, \dots, t_N$  are different. Since  $x_\alpha \neq 0 \pmod p$  this implies that we cannot find  $N$  different  $t_1, \dots, t_N$  so  $\sum_{i=1}^N t_i^{i-1}x_i = 0 \pmod p$ . When choosing  $t$  at random there is at least probability  $1 - \frac{N-1}{p}$  that  $\sum_{i=1}^N t^{i-1}x_i \neq 0 \pmod p$ . Since  $p$  is superpolynomial, this probability is negligible. The same argument holds for all other prime factors in  $n$ , so with overwhelming probability  $h$  is a generator of  $G$ . □

## References

BB04. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

- BF03. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. *SIAM J. Comput.* 32(3), 586–615 (2003)
- BGN05. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Kilian, J. (ed.) *TCC 2005*. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
- BKM06. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) *TCC 2006*. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006)
- Boy07. Boyen, X.: Mesh signatures. In: *Advances in Cryptology—EUROCRYPT 2007*. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007), Available at <http://www.cs.stanford.edu/~xb/eurocrypt07b/>
- BW06. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) *EUROCRYPT 2006*. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
- CWLY06. Chow, S.S.M., Wei, V.K., Liu, J.K., Yuen, T.H.: Ring Signatures without Random Oracles. In: *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, Taipei, Taiwan. Computer and Communications Security*, pp. 297–302. ACM Press, New York (2006)
- DKNS04. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004)
- GOS06. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero-knowledge for np. In: Vaudenay, S. (ed.) *EUROCRYPT 2006*. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
- GS06. Groth, J., Sahai, A.: Efficient non-interactive proofs for bilinear groups. Manuscript (2006)
- JSI96. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) *EUROCRYPT 1996*. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
- Len87. Lenstra, H.W.: Factoring integers with elliptic curves. *Annals of Mathematics* 126, 649–673 (1987)
- Nao02. Naor, M.: Deniable ring authentication. In: Yung, M. (ed.) *CRYPTO 2002*. LNCS, vol. 2442, pp. 481–498. Springer, Heidelberg (2002)
- RST06. Rivest, R., Shamir, A., Tauman, Y.: How to leak a secret: Theory and applications of ring signatures. In: *Essays in Memory of Shimon Even* (2006)
- SW06. Shacham, H., Waters, B.: Efficient ring signatures without random oracles (2006), Available at <http://eprint.iacr.org/2006/289.pdf>

# Balanced Families of Perfect Hash Functions and Their Applications

Noga Alon<sup>1,\*</sup> and Shai Gutner<sup>2,\*\*</sup>

<sup>1</sup> Schools of Mathematics and Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel

noga@math.tau.ac.il

<sup>2</sup> School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel  
gutner@tau.ac.il

**Abstract.** The construction of perfect hash functions is a well-studied topic. In this paper, this concept is generalized with the following definition. We say that a family of functions from  $[n]$  to  $[k]$  is a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions if for every  $S \subseteq [n]$ ,  $|S| = k$ , the number of functions that are 1-1 on  $S$  is between  $T/\delta$  and  $\delta T$  for some constant  $T > 0$ . The standard definition of a family of perfect hash functions requires that there will be at least one function that is 1-1 on  $S$ , for each  $S$  of size  $k$ . In the new notion of balanced families, we require the number of 1-1 functions to be almost the same (taking  $\delta$  to be close to 1) for every such  $S$ . Our main result is that for any constant  $\delta > 1$ , a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions of size  $2^{O(k \log \log k)} \log n$  can be constructed in time  $2^{O(k \log \log k)} n \log n$ . Using the technique of color-coding we can apply our explicit constructions to devise approximation algorithms for various counting problems in graphs. In particular, we exhibit a deterministic polynomial time algorithm for approximating both the number of simple paths of length  $k$  and the number of simple cycles of size  $k$  for any  $k \leq O(\frac{\log n}{\log \log \log n})$  in a graph with  $n$  vertices. The approximation is up to any fixed desirable relative error.

**Keywords:** approximate counting of subgraphs, color-coding, perfect hashing.

## 1 Introduction

This paper deals with explicit constructions of balanced families of perfect hash functions. The topic of perfect hash functions has been widely studied under the more general framework of  $k$ -restriction problems (see, e.g., [3], [13]). These problems have an existential nature of requiring a set of conditions to hold at least once for any choice of  $k$  elements out of the problem domain. We generalize the definition of perfect hash functions, and introduce a new, simple, and

---

\* Research supported in part by a grant from the Israel Science Foundation, and by the Hermann Minkowski Minerva Center for Geometry at Tel Aviv University.

\*\* This paper forms part of a Ph.D. thesis written by the author under the supervision of Prof. N. Alon and Prof. Y. Azar in Tel Aviv University.

yet useful notion which we call balanced families of perfect hash functions. The purpose of our new definition is to incorporate more structure into the constructions. Our explicit constructions together with the method of color-coding from [5] are applied for problems of approximating the number of times that some fixed subgraph appears within a large graph. We focus on counting simple paths and simple cycles. Recently, the method of color-coding has found interesting applications in computational biology ([17], [18], [19], [12]), specifically in detecting signaling pathways within protein interaction. This problem is formalized using an undirected edge-weighted graph, where the task is to find a minimum weight path of length  $k$ . The application of our results in this case is for approximating deterministically the number of minimum weight paths of length  $k$ .

**Perfect Hash Functions.** An  $(n, k)$ -family of perfect hash functions is a family of functions from  $[n]$  to  $[k]$  such that for every  $S \subseteq [n]$ ,  $|S| = k$ , there exists a function in the family that is 1-1 on  $S$ . There is an extensive literature dealing with explicit constructions of perfect hash functions. The construction described in [5] (following [11] and [16]) is of size  $2^{O(k)} \log n$ . The best known explicit construction is of size  $e^k k^{O(\log k)} \log n$ , which closely matches the known lower bound of  $\Omega(e^k \log n / \sqrt{k})$  [15].

**Finding and Counting Paths and Cycles.** The foundations for the graph algorithms presented in this paper have been laid in [5]. Two main randomized algorithms are presented there, as follows. A simple directed or undirected path of length  $k - 1$  in a graph  $G = (V, E)$  that contains such a path can be found in  $2^{O(k)}|E|$  expected time in the directed case and in  $2^{O(k)}|V|$  expected time in the undirected case. A simple directed or undirected cycle of size  $k$  in a graph  $G = (V, E)$  that contains such a cycle can be found in either  $2^{O(k)}|V||E|$  or  $2^{O(k)}|V|^\omega$  expected time, where  $\omega < 2.376$  is the exponent of matrix multiplication. The derandomization of these algorithms incur an extra  $\log |V|$  factor. As for the case of even cycles, it is shown in [20] that for every fixed  $k \geq 2$ , there is an  $O(|V|^2)$  algorithm for finding a simple cycle of size  $2k$  in an undirected graph. Improved algorithms for detecting given length cycles have been presented in [6] and [21]. An interesting result from [6], related to the questions addressed in the present paper, is an  $O(|V|^\omega)$  algorithm for counting the number of cycles of size at most 7. Flum and Grohe proved that the problem of counting *exactly* the number of paths and cycles of length  $k$  in both directed and undirected graphs, parameterized by  $k$ , is  $\#W[1]$ -complete [10]. Their result implies that most likely there is no  $f(k) \cdot n^c$ -algorithm for counting the precise number of paths or cycles of length  $k$  in a graph of size  $n$  for any computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  and constant  $c$ . This suggests the problem of approximating these quantities. Arvind and Raman obtained a *randomized* fixed-parameter tractable algorithm to approximately count the number of copies of a fixed subgraph with bounded treewidth within a large graph [7]. We settle in the affirmative the open question they raise concerning the existence of a *deterministic* approximate counting algorithm for this problem. For simplicity, we give algorithms for approximately counting paths and cycles. These results can be easily extended to the problem

of approximately counting bounded treewidth subgraphs, combining the same approach with the method of [5]. The main new ingredient in our deterministic algorithms is the application of balanced families of perfect hash functions- a combinatorial notion introduced here which, while simple, appears to be very useful.

**Balanced Families of Perfect Hash Functions.** We say that a family of functions from  $[n]$  to  $[k]$  is a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions if for every  $S \subseteq [n]$ ,  $|S| = k$ , the number of functions that are 1-1 on  $S$  is between  $T/\delta$  and  $\delta T$  for some constant  $T > 0$ . Balanced families of perfect hash functions are a natural generalization of the usual concept of perfect hash functions. To assist with our explicit constructions, we define also the even more generalized notion of balanced splitters. (See section 2 for the definition. This is a generalization of an ordinary splitter defined in [15].)

**Our Results.** The main focus of the paper is on explicit constructions of balanced families of perfect hash functions and their applications. First, we give non-constructive upper bounds on the size of different types of balanced splitters. Then, we compare these bounds with those achieved by constructive algorithms. Our main result is an explicit construction, for every  $1 < \delta \leq 2$ , of a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions of size  $2^{O(k \log \log k)}(\delta - 1)^{-O(\log k)} \log n$ . The running time of the procedure that provides the construction is  $2^{O(k \log \log k)}(\delta - 1)^{-O(\log k)} n \log n + (\delta - 1)^{-O(k/\log k)}$ .

Constructions of balanced families of perfect hash functions can be applied to various counting problems in graphs. In particular, we describe deterministic algorithms that approximate the number of times that a small subgraph appears within a large graph. The approximation is always up to some multiplicative factor, that can be made arbitrarily close to 1. For any  $1 < \delta \leq 2$ , the number of simple paths of length  $k - 1$  in a graph  $G = (V, E)$  can be approximated up to a multiplicative factor of  $\delta$  in time  $2^{O(k \log \log k)}(\delta - 1)^{-O(\log k)} |E| \log |V| + (\delta - 1)^{-O(k/\log k)}$ . The number of simple cycles of size  $k$  can be approximated up to a multiplicative factor of  $\delta$  in time  $2^{O(k \log \log k)}(\delta - 1)^{-O(\log k)} |E| |V| \log |V| + (\delta - 1)^{-O(k/\log k)}$ .

**Techniques.** We use probabilistic arguments in order to prove the existence of different types of small size balanced splitters (whose precise definition is given in the next section). To construct a balanced splitter, a natural randomized algorithm is to choose a large enough number of independent random functions. We show that in some cases, the method of conditional probabilities, when applied on a proper choice of a potential function, can derandomize this process in an efficient way. Constructions of small probability spaces that admit  $k$ -wise independent random variables are also a natural tool for achieving good splitting properties. The use of error correcting codes is shown to be useful when we want to find a family of functions from  $[n]$  to  $[l]$ , where  $l$  is much bigger than  $k^2$ , such that for every  $S \subseteq [n]$ ,  $|S| = k$ , almost all of the functions should be 1-1 on  $S$ . Balanced splitters can be composed in different ways and our main construction is achieved by composing three types of splitters. We apply the

explicit constructions of balanced families of perfect hash functions together with the color-coding technique to get our approximate counting algorithms.

## 2 Balanced Families of Perfect Hash Functions

In this section we formally define the new notions of balanced families of perfect hash functions and balanced splitters. Here are a few basics first. Denote by  $[n]$  the set  $\{1, \dots, n\}$ . For any  $k, 1 \leq k \leq n$ , the family of  $k$ -sized subsets of  $[n]$  is denoted by  $\binom{[n]}{k}$ . We denote by  $k \bmod l$  the unique integer  $0 \leq r < l$  for which  $k = ql + r$ , for some integer  $q$ . We now introduce the new notion of balanced families of perfect hash functions.

**Definition 1.** *Suppose that  $1 \leq k \leq n$  and  $\delta \geq 1$ . We say that a family of functions from  $[n]$  to  $[k]$  is a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions if there exists a constant real number  $T > 0$ , such that for every  $S \in \binom{[n]}{k}$ , the number of functions that are 1-1 on  $S$ , which we denote by  $\text{inj}(S)$ , satisfies the relation  $T/\delta \leq \text{inj}(S) \leq \delta T$ .*

The following definition generalizes both the last definition and the definition of a splitter from [15].

**Definition 2.** *Suppose that  $1 \leq k \leq n$  and  $\delta \geq 1$ , and let  $H$  be a family of functions from  $[n]$  to  $[l]$ . For a set  $S \in \binom{[n]}{k}$  we denote by  $\text{split}(S)$  the number of functions  $h \in H$  that split  $S$  into equal-sized parts  $h^{-1}(j) \cap S, j = 1, \dots, l$ . In case  $l$  does not divide  $k$  we separate between two cases. If  $k \leq l$ , then  $\text{split}(S)$  is defined to be the number of functions that are 1-1 on  $S$ . Otherwise,  $k > l$  and we require the first  $k \bmod l$  parts to be of size  $\lceil k/l \rceil$  and the remaining parts to be of size  $\lfloor k/l \rfloor$ . We say that  $H$  is a  $\delta$ -balanced  $(n, k, l)$ -splitter if there exists a constant real number  $T > 0$ , such that for every  $S \in \binom{[n]}{k}$  we have  $T/\delta \leq \text{split}(S) \leq \delta T$ .*

The definitions of balanced families of perfect hash functions and balanced splitters given above enable us to state the following easy composition lemmas.

**Lemma 1.** *For any  $k < l$ , let  $H$  be an explicit  $\delta$ -balanced  $(n, k, l)$ -splitter of size  $N$  and let  $G$  be an explicit  $\gamma$ -balanced  $(l, k)$ -family of perfect hash functions of size  $M$ . We can use  $H$  and  $G$  to get an explicit  $\delta\gamma$ -balanced  $(n, k)$ -family of perfect hash functions of size  $NM$ .*

*Proof.* We compose every function of  $H$  with every function of  $G$  and get the needed result. □

**Lemma 2.** *For any  $k > l$ , let  $H$  be an explicit  $\delta$ -balanced  $(n, k, l)$ -splitter of size  $N$ . For every  $j, j = 1, \dots, l$ , let  $G_j$  be an explicit  $\gamma_j$ -balanced  $(n, k_j)$ -family of perfect hash functions of size  $M_j$ , where  $k_j = \lceil k/l \rceil$  for every  $j \leq k \bmod l$  and  $k_j = \lfloor k/l \rfloor$  otherwise. We can use these constructions to get an explicit  $(\delta \prod_{j=1}^l \gamma_j)$ -balanced  $(n, k)$ -family of perfect hash functions of size  $N \prod_{j=1}^l M_j$ .*

*Proof.* We divide the set  $[k]$  into  $l$  disjoint intervals  $I_1, \dots, I_l$ , where the size of  $I_j$  is  $k_j$  for every  $j = 1, \dots, l$ . We think of  $G_j$  as a family of functions from  $[n]$  to  $I_j$ . For every combination of  $h \in H$  and  $g_j \in G_j, j = 1, \dots, l$ , we create a new function that maps an element  $x \in [n]$  to  $g_{h(x)}(x)$ .  $\square$

### 3 Probabilistic Constructions

We will use the following two claims: a variant of the Chernoff bound (c.f., e.g., [4]) and Robbins' formula [9] (a tight version of Stirling's formula).

*Claim.* Let  $Y$  be the sum of mutually independent indicator random variables,  $\mu = E[Y]$ . For all  $1 \leq \delta \leq 2$ ,

$$Pr\left[\frac{\mu}{\delta} \leq Y \leq \delta\mu\right] > 1 - 2e^{-(\delta-1)^2\mu/8}.$$

*Claim.* For every integer  $n \geq 1$ ,

$$\sqrt{2\pi n}^{n+1/2} e^{-n+1/(12n+1)} < n! < \sqrt{2\pi n}^{n+1/2} e^{-n+1/(12n)}.$$

Now we state the results for  $\delta$ -balanced  $(n, k, l)$ -splitters of the three types:  $k = l, k < l$  and  $k > l$ .

**Theorem 1.** *For any  $1 < \delta \leq 2$ , there exists a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions of size  $O\left(\frac{e^k \sqrt{k \log n}}{(\delta-1)^2}\right)$ .*

*Proof.* (sketch) Set  $p = k!/k^k$  and  $M = \lceil \frac{8(k \ln n + 1)}{p(\delta-1)^2} \rceil$ . We choose  $M$  independent random functions. For a specific set  $S \in \binom{[n]}{k}$ , the expected number of functions that are 1-1 on  $S$  is exactly  $pM$ . By the Chernoff bound, the probability that for at least one set  $S \in \binom{[n]}{k}$ , the number of functions that are 1-1 on  $S$  will not be as needed is at most

$$\binom{n}{k} 2e^{-(\delta-1)^2 pM/8} \leq 2 \binom{n}{k} e^{-(k \ln n + 1)} < 1. \quad \square$$

**Theorem 2.** *For any  $k < l$  and  $1 < \delta \leq 2$ , there exists a  $\delta$ -balanced  $(n, k, l)$ -splitter of size  $O\left(\frac{e^{k^2/l} k \log n}{(\delta-1)^2}\right)$ .*

*Proof.* (sketch) We set  $p = \frac{l!}{(l-k)l^k}$  and  $M = \lceil \frac{8(k \ln n + 1)}{p(\delta-1)^2} \rceil$ . Using Robbins' formula, we get

$$\frac{1}{p} \leq e^{k+1/12} \left(1 - \frac{k}{l}\right)^{l-k+1/2} \leq e^{k+1/12} e^{-\frac{k}{l}(l-k+1/2)} = e^{\frac{k^2-k/2}{l} + 1/12}.$$

We choose  $M$  independent random functions and proceed as in the proof of Theorem 1.  $\square$



For the case  $k > l$ , the probabilistic arguments from [15] can be generalized to prove existence of balanced  $(n, k, l)$ -splitters. Here we focus on the special case of balanced  $(n, k, 2)$ -splitters, which will be of interest later.

**Theorem 3.** *For any  $k \geq 2$  and  $1 < \delta \leq 2$ , there exists a  $\delta$ -balanced  $(n, k, 2)$ -splitter of size  $O(\frac{k\sqrt{k \log n}}{(\delta-1)^2})$ .*

*Proof.* (sketch) Set  $M = \lceil \frac{8(k \ln n + 1)}{p(\delta-1)^2} \rceil$ , where  $p$  denotes the probability to get the needed split in a random function. It follows easily from Robbins' formula that  $1/p = O(\sqrt{k})$ . We choose  $M$  independent random functions and proceed as in the proof of Theorem 1. □

### 4 Explicit Constructions

In this paper, we use the term explicit construction for an algorithm that lists all the elements of the required family of functions in time which is polynomial in the total size of the functions. For a discussion on other definitions for this term, the reader is referred to [15]. We state our results for  $\delta$ -balanced  $(n, k, l)$ -splitters of the three types:  $k = l$ ,  $k < l$  and  $k > l$ .

**Theorem 4.** *For any  $1 < \delta \leq 2$ , a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions of size  $O(\frac{e^k \sqrt{k \log n}}{(\delta-1)^2})$  can be constructed deterministically within time  $\binom{n}{k} \frac{e^k k^{O(1)} n \log n}{(\delta-1)^2}$ .*

*Proof.* We set  $p = k!/k^k$  and  $M = \lceil \frac{16(k \ln n + 1)}{p(\delta-1)^2} \rceil$ . Denote  $\lambda = (\delta - 1)/4$ , so obviously  $0 < \lambda \leq 1/4$ . Consider a choice of  $M$  independent random functions from  $[n]$  to  $[k]$ . This choice will be derandomized in the course of the algorithm. For every  $S \in \binom{[n]}{k}$ , we define  $X_S = \sum_{i=1}^M X_{S,i}$ , where  $X_{S,i}$  is the indicator random variable that is equal to 1 iff the  $i$ th function is 1-1 on  $S$ . Consider the following potential function:

$$\Phi = \sum_{S \in \binom{[n]}{k}} e^{\lambda(X_S - pM)} + e^{\lambda(pM - X_S)}.$$

Its expectation can be calculated as follows:

$$\begin{aligned} E[\Phi] &= \binom{n}{k} (e^{-\lambda pM} \prod_{i=1}^M E[e^{\lambda X_{S,i}}] + e^{\lambda pM} \prod_{i=1}^M E[e^{-\lambda X_{S,i}}]) = \\ &= \binom{n}{k} (e^{-\lambda pM} [pe^\lambda + (1-p)]^M + e^{\lambda pM} [pe^{-\lambda} + (1-p)]^M). \end{aligned}$$

We now give an upper bound for  $E[\Phi]$ . Since  $1 + u \leq e^u$  for all  $u$  and  $e^{-u} \leq 1 - u + u^2/2$  for all  $u \geq 0$ , we get that  $pe^{-\lambda} + (1-p) \leq e^{p(e^{-\lambda}-1)} \leq e^{p(-\lambda+\lambda^2/2)}$ .

Define  $\epsilon = e^\lambda - 1$ , that is  $\lambda = \ln(1 + \epsilon)$ . Thus  $pe^\lambda + (1 - p) = 1 + \epsilon p \leq e^{\epsilon p}$ . This implies that

$$E[\Phi] \leq n^k \left( \left( \frac{e^\epsilon}{1 + \epsilon} \right)^{pM} + e^{\lambda^2 pM/2} \right).$$

Since  $e^u \leq 1 + u + u^2$  for all  $0 \leq u \leq 1$ , we have that  $\frac{e^\epsilon}{1+\epsilon} = e^{e^\lambda - 1 - \lambda} \leq e^{\lambda^2}$ . We conclude that

$$E[\Phi] \leq 2n^k e^{\lambda^2 pM} \leq e^{2(k \ln n + 1)}.$$

We now describe a deterministic algorithm for finding  $M$  functions, so that  $E[\Phi]$  will still obey the last upper bound. This is performed using the method of conditional probabilities (c.f., e.g., [4], chapter 15). The algorithm will have  $M$  phases, where each phase will consist of  $n$  steps. In step  $i$  of phase  $j$  the algorithm will determine the  $i$ th value of the  $j$ th function. Out of the  $k$  possible values, we greedily choose the value that will decrease  $E[\Phi]$  as much as possible. We note that at any specific step of the algorithm, the exact value of the conditional expectation of the potential function can be easily computed in time  $\binom{n}{k} k^{O(1)}$ .

After all the  $M$  functions have been determined, every set  $S \in \binom{[n]}{k}$  satisfies the following:

$$e^{\lambda(X_S - pM)} + e^{\lambda(pM - X_S)} \leq e^{2(k \ln n + 1)}.$$

This implies that

$$-2(k \ln n + 1) \leq \lambda(X_S - pM) \leq 2(k \ln n + 1).$$

Recall that  $\lambda = (\delta - 1)/4$ , and therefore

$$\left( 1 - \frac{8(k \ln n + 1)}{(\delta - 1)pM} \right) pM \leq X_S \leq \left( 1 + \frac{8(k \ln n + 1)}{(\delta - 1)pM} \right) pM.$$

Plugging in the values of  $M$  and  $p$  we get that

$$\left( 1 - \frac{\delta - 1}{2} \right) pM \leq X_S \leq \left( 1 + \frac{\delta - 1}{2} \right) pM.$$

Using the fact that  $1/u \leq 1 - (u - 1)/2$  for all  $1 \leq u \leq 2$ , we get the desired result

$$pM/\delta \leq X_S \leq \delta pM.$$

□

**Theorem 5.** For any  $1 < \delta \leq 2$ , a  $\delta$ -balanced  $(n, k, \lceil \frac{2k^2}{\delta - 1} \rceil)$ -splitter of size  $\frac{k^{O(1)} \log n}{(\delta - 1)^{O(1)}}$  can be constructed in time  $\frac{k^{O(1)} n \log n}{(\delta - 1)^{O(1)}}$ .

*Proof.* Denote  $q = \lceil \frac{2k^2}{\delta - 1} \rceil$ . Consider an explicit construction of an error correcting code with  $n$  codewords over alphabet  $[q]$  whose normalized Hamming distance is at least  $1 - \frac{2}{q}$ . Such explicit codes of length  $O(q^2 \log n)$  exist [11]. Now let every index of the code corresponds to a function from  $[n]$  to  $[q]$ . If we denote by  $M$

the length of the code, which is in fact the size of the splitter, then for every  $S \in \binom{[n]}{k}$ , the number of good splits is at least

$$\left(1 - \binom{k}{2} \frac{2}{q}\right)M \geq \left(1 - \frac{\delta - 1}{2}\right)M \geq M/\delta,$$

where the last inequality follows from the fact that  $1 - (u - 1)/2 \geq 1/u$  for all  $1 \leq u \leq 2$ . □

For our next construction we use small probability spaces that support a sequence of almost  $k$ -size independent random variables. A sequence  $X_1, \dots, X_n$  of random Boolean variables is  $(\epsilon, k)$ -independent if for any  $k$  positions  $i_1 < \dots < i_k$  and any  $k$  bits  $\alpha_1, \dots, \alpha_k$  we have

$$|Pr[X_{i_1} = \alpha_1, \dots, X_{i_k} = \alpha_k] - 2^{-k}| < \epsilon.$$

It is known ([14],[2],[1]) that sample spaces of size  $2^{O(k+\log \frac{1}{\epsilon})} \log n$  that support  $n$  random variables that are  $(\epsilon, k)$ -independent can be constructed in time  $2^{O(k+\log \frac{1}{\epsilon})} n \log n$ .

**Theorem 6.** *For any  $k \geq l$  and  $1 < \delta \leq 2$ , a  $\delta$ -balanced  $(n, k, l)$ -splitter of size  $2^{O(k \log l - \log(\delta - 1))} \log n$  can be constructed in time  $2^{O(k \log l - \log(\delta - 1))} n \log n$ .*

*Proof.* We use an explicit probability space of size  $2^{O(k \log l - \log(\delta - 1))} \log n$  that supports  $n \lceil \log_2 l \rceil$  random variables that are  $(\epsilon, k \lceil \log_2 l \rceil)$ -independent where  $\epsilon = 2^{-k \lceil \log_2 l \rceil - 1} (\delta - 1)$ . We attach  $\lceil \log_2 l \rceil$  random variables to each element of  $[n]$ , thereby assigning it a value from  $[2^{\lceil \log_2 l \rceil}]$ . In case  $l$  is not a power of 2, all elements of  $[2^{\lceil \log_2 l \rceil}] - [l]$  can be mapped to  $[l]$  by some arbitrary fixed function. It follows from the construction that there exists a constant  $T > 0$  so that for every  $S \in \binom{[n]}{k}$ , the number of good splits satisfies

$$\frac{T}{\delta} \leq \left(1 - \frac{\delta - 1}{2}\right)T \leq split(S) \leq \left(1 + \frac{\delta - 1}{2}\right)T \leq \delta T.$$

□

**Corollary 1.** *For any fixed  $c > 0$ , a  $(1 + c^{-k})$ -balanced  $(n, k, 2)$ -splitter of size  $2^{O(k)} \log n$  can be constructed in time  $2^{O(k)} n \log n$ .*

Setting  $l = k$  in Theorem 6, we get that a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions of size  $2^{O(k \log k - \log(\delta - 1))} \log n$  can be constructed in time  $2^{O(k \log k - \log(\delta - 1))} n \log n$ . Note that if  $k$  is small enough with respect to  $n$ , say  $k = O(\log n / \log \log n)$ , then for any fixed  $1 < \delta \leq 2$ , this already gives a family of functions of size polynomial in  $n$ . We improve upon this last result in the following Theorem, which is our main construction.

**Theorem 7.** *For  $1 < \delta \leq 2$ , a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions of size  $\frac{2^{O(k \log \log k)}}{(\delta - 1)^{O(\log k)}} \log n$  can be constructed in time  $\frac{2^{O(k \log \log k)}}{(\delta - 1)^{O(\log k)}} n \log n + (\delta - 1)^{-O(k/\log k)}$ . In particular, for any fixed  $1 < \delta \leq 2$ , the size is  $2^{O(k \log \log k)} \log n$  and the time is  $2^{O(k \log \log k)} n \log n$ .*

*Proof.* (sketch) Denote  $l = \lceil \log_2 k \rceil$ ,  $\delta' = \delta^{1/3}$ ,  $\delta'' = \delta^{1/(3l)}$ , and  $q = \lceil \frac{2k^2}{\delta' - 1} \rceil$ . Let  $H$  be a  $\delta'$ -balanced  $(q, k, l)$ -splitter of size  $2^{O(k \log \log k)}(\delta' - 1)^{-O(1)}$  constructed using Theorem 6. For every  $j$ ,  $j = 1, \dots, l$ , let  $B_j$  be a  $\delta''$ -balanced  $(q, k_j)$ -family of perfect hash functions of size  $O(e^{k/\log k} k)(\delta'' - 1)^{-O(1)}$  constructed using Theorem 4, where  $k_j = \lceil k/l \rceil$  for every  $j \leq k \bmod l$  and  $k_j = \lfloor k/l \rfloor$  otherwise. Using Lemma 2 for composing  $H$  and  $\{B_j\}_{j=1}^l$ , we get a  $\delta'^2$ -balanced  $(q, k)$ -family  $D'$  of perfect hash functions.

Now let  $D''$  be a  $\delta'$ -balanced  $(n, k, q)$ -splitter of size  $k^{O(1)}(\delta' - 1)^{-O(1)} \log n$  constructed using Theorem 5. Using Lemma 1 for composing  $D'$  and  $D''$ , we get a  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions, as needed. Note that for calculating the size of each  $B_j$ , we use the fact that  $e^{u/2} \leq 1 + u \leq e^u$  for all  $0 \leq u \leq 1$ , and get the following:

$$\delta'' - 1 = (1 + (\delta - 1))^{\frac{1}{3l}} - 1 \geq e^{\frac{\delta - 1}{6l}} - 1 \geq \frac{\delta - 1}{6l}.$$

The time needed to construct each  $B_j$  is  $2^{O(k)}(\delta' - 1)^{-O(k/\log k)}$ . The  $2^{O(k)}$  term is omitted in the final result, as it is negligible in respect to the other terms.  $\square$

## 5 Approximate Counting of Paths and Cycles

We now state what it means for an algorithm to approximate a counting problem.

**Definition 3.** *We say that an algorithm approximates a counting problem by a multiplicative factor  $\delta \geq 1$  if for every input  $x$ , the output  $ALG(x)$  of the algorithm satisfies  $N(x)/\delta \leq ALG(x) \leq \delta N(x)$ , where  $N(x)$  is the exact output of the counting problem for input  $x$ .*

The technique of color-coding is used for approximate counting of paths and cycles. Let  $G = (V, E)$  be a directed or undirected graph. In our algorithms we will use constructions of balanced  $(|V|, k)$ -families of perfect hash functions. Each such function defines a coloring of the vertices of the graph. A path is said to be *colorful* if each vertex on it is colored by a distinct color. Our goal is to count the exact number of colorful paths in each of these colorings.

**Theorem 8.** *For any  $1 < \delta \leq 2$ , the number of simple (directed or undirected) paths of length  $k - 1$  in a (directed or undirected) graph  $G = (V, E)$  can be approximated up to a multiplicative factor of  $\delta$  in time  $\frac{2^{O(k \log \log k)}}{(\delta - 1)^{O(\log k)}} |E| \log |V| + (\delta - 1)^{-O(k/\log k)}$ .*

*Proof.* (sketch) We use the  $\delta$ -balanced  $(|V|, k)$ -family of perfect hash functions constructed using Theorem 7. Each function of the family defines a coloring of the vertices in  $k$  colors. We know that there exists a constant  $T > 0$ , so that for each set  $S \subseteq V$  of  $k$  vertices, the number of functions that are 1-1 on  $S$  is between  $T/\delta$  and  $\delta T$ . The exact value of  $T$  can be easily calculated in all of our explicit constructions.

For each coloring, we use a dynamic programming approach in order to calculate the exact number of colorful paths. We do this in  $k$  phases. In the  $i$ th phase, for each vertex  $v \in V$  and for each subset  $C \subseteq \{1, \dots, k\}$  of  $i$  colors, we calculate the number of colorful paths of length  $i - 1$  that end at  $v$  and use the colors of  $C$ . To do so, for every edge  $(u, v) \in E$ , we check whether it can be the last edge of a colorful path of length  $i - 1$  ending at either  $u$  or  $v$ . Its contribution to the number of paths of length  $i - 1$  is calculated using our knowledge on the number of paths of length  $i - 2$ . The initialization of phase 1 is easy and after performing phase  $k$  we know the exact number of paths of length  $k - 1$  that end at each vertex  $v \in V$ . The time to process each coloring is therefore  $2^{O(k)}|E|$ .

We sum the results over all colorings and all ending vertices  $v \in V$ . The result is divided by  $T$ . In case the graph is undirected, we further divide by 2. This is guaranteed to be the needed approximation.  $\square$

**Theorem 9.** *For any  $1 < \delta \leq 2$ , the number of simple (directed or undirected) cycles of size  $k$  in a (directed or undirected) graph  $G = (V, E)$  can be approximated up to a multiplicative factor of  $\delta$  in time  $\frac{2^{O(k \log \log k)}}{(\delta - 1)^{O(\log k)}}|E||V| \log |V| + (\delta - 1)^{-O(k/\log k)}$ .*

*Proof.* (sketch) We use the  $\delta$ -balanced  $(|V|, k)$ -family of perfect hash functions constructed using Theorem 7. For every set  $S$  of  $k$  vertices, the number of functions that are 1-1 on  $S$  is between  $T/\delta$  and  $\delta T$ . Every function defines a coloring and for each such coloring we proceed as follows. For every vertex  $s \in V$  we run the algorithm described in the proof of Theorem 8 in order to calculate for each vertex  $v \in V$  the exact number of colorful paths of length  $k - 1$  from  $s$  to  $v$ . In case there is an edge  $(v, s)$  that completes a cycle, we add the result to our count.

We sum the results over all the colorings and all pairs of vertices  $s$  and  $v$  as described above. The result is divided by  $kT$ . In case the graph is undirected, we further divide by 2. The needed approximation is achieved.  $\square$

**Corollary 2.** *For any constant  $c > 0$ , there is a deterministic polynomial time algorithm for approximating both the number of simple paths of length  $k$  and the number of simple cycles of size  $k$  for every  $k \leq O(\frac{\log n}{\log \log \log n})$  in a graph with  $n$  vertices, where the approximation is up to a multiplicative factor of  $1 + (\ln \ln n)^{-c \ln \ln n}$ .*

## 6 Concluding Remarks

- An interesting open problem is whether for every fixed  $\delta > 1$ , there exists an explicit  $\delta$ -balanced  $(n, k)$ -family of perfect hash functions of size  $2^{O(k)} \log n$ . The key ingredient needed is an improved construction of balanced  $(n, k, 2)$ -splitters. Such splitters can be applied successively to get the balanced  $(n, k, \lceil \log_2 k \rceil)$ -splitter needed in Theorem 7. It seems that the constructions presented in [2] could be good candidates for balanced  $(n, k, 2)$ -splitters, although the Fourier analysis in this case (along the lines of [8]) seems elusive.

- Other algorithms from [5] can be generalized to deal with counting problems. In particular it is possible to combine our approach here with the ideas of [5] based on fast matrix multiplication in order to approximate the number of cycles of a given length. Given a forest  $F$  on  $k$  vertices, the number of subgraphs of  $G$  isomorphic to  $F$  can be approximated using a recursive algorithm similar to the one in [5]. For a weighted graph, we can approximate, for example, both the number of minimum (maximum) weight paths of length  $k - 1$  and the number of minimum (maximum) weight cycles of size  $k$ . Finally, all the results can be readily extended from paths and cycles to arbitrary small subgraphs of bounded tree-width. We omit the details.
- In the definition of a balanced  $(n, k)$ -family of perfect hash functions, there is some constant  $T > 0$ , such that for every  $S \subseteq [n]$ ,  $|S| = k$ , the number of functions that are 1-1 on  $S$  is close to  $T$ . We note that the value of  $T$  need not be equal to the expected number of 1-1 functions on a set of size  $k$ , for the case that the functions were chosen independently according to a uniform distribution. For example, the value of  $T$  in the construction of Theorem 7 is not even asymptotically equal to what one would expect in a uniform distribution.

## References

1. Alon, N., Bruck, J., Naor, J., Naor, M., Roth, R.M.: Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory* 38(2), 509 (1992)
2. Alon, N., Goldreich, O., Håstad, J., Peralta, R.: Simple construction of almost  $k$ -wise independent random variables. *Random Struct. Algorithms* 3(3), 289–304 (1992)
3. Alon, N., Moshkovitz, D., Safra, S.: Algorithmic construction of sets for  $k$ -restrictions. *ACM Transactions on Algorithms* 2(2), 153–177 (2006)
4. Alon, N., Spencer, J.H.: *The Probabilistic Method*, 2nd edn. Wiley, Chichester (2000)
5. Alon, N., Yuster, R., Zwick, U.: Color-coding. *Journal of the ACM* 42(4), 844–856 (1995)
6. Alon, N., Yuster, R., Zwick, U.: Finding and counting given length cycles. *Algorithmica* 17(3), 209–223 (1997)
7. Arvind, V., Raman, V.: Approximation algorithms for some parameterized counting problems. In: Bose, P., Morin, P. (eds.) *ISAAC 2002*. LNCS, vol. 2518, pp. 453–464. Springer, Heidelberg (2002)
8. Azar, Y., Motwani, R., Naor, J.: Approximating probability distributions using small sample spaces. *Combinatorica* 18(2), 151–171 (1998)
9. Feller, W.: *An introduction to probability theory and its applications*, 3rd edn., vol. I. Wiley, Chichester (1968)
10. Flum, J., Grohe, M.: The parameterized complexity of counting problems. *SIAM Journal on Computing* 33(4), 892–922 (2004)
11. Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a sparse table with  $O(1)$  worst case access time. *Journal of the ACM* 31(3), 538–544 (1984)

12. Hüffner, F., Wernicke, S., Zichner, T.: Algorithm engineering for color-coding to facilitate signaling pathway detection. In: Sankoff, D., Wang, L., Chin, F. (eds.) APBC 2007. Proceedings of 5th Asia-Pacific Bioinformatics Conference, Hong Kong, China, January 15-17, 2007. *Advances in Bioinformatics and Computational Biology*, vol. 5, pp. 277–286. Imperial College Press, Imperial (2007)
13. Koller, D., Megiddo, N.: Constructing small sample spaces satisfying given constraints. *SIAM Journal on Discrete Mathematics* 7(2), 260–274 (1994)
14. Naor, J., Naor, M.: Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing* 22(4), 838–856 (1993)
15. Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In: 36th Annual Symposium on Foundations of Computer Science, pp. 182–191 (1995)
16. Schmidt, J.P., Siegel, A.: The spatial complexity of oblivious  $k$ -probe hash functions. *SIAM Journal on Computing* 19(5), 775–786 (1990)
17. Scott, J., Ideker, T., Karp, R.M., Sharan, R.: Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology* 13(2), 133–144 (2006)
18. Sharan, R., Ideker, T.: Modeling cellular machinery through biological network comparison. *Nature Biotechnology* 24(4), 427–433 (2006)
19. Shlomi, T., Segal, D., Ruppin, E., Sharan, R.: QPath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics* 7, 199 (2006)
20. Yuster, R., Zwick, U.: Finding even cycles even faster. *SIAM Journal on Discrete Mathematics* 10(2), 209–222 (1997)
21. Yuster, R., Zwick, U.: Detecting short directed cycles using rectangular matrix multiplication and dynamic programming. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 254–260. ACM Press, New York (2004)

# An Exponential Improvement on the MST Heuristic for Minimum Energy Broadcasting in Ad Hoc Wireless Networks\*

Ioannis Caragiannis<sup>1</sup>, Michele Flammini<sup>2</sup>, and Luca Moscardelli<sup>2</sup>

<sup>1</sup> Research Academic Computer Technology Institute and  
Dept. of Computer Engineering and Informatics  
University of Patras, 26500 Rio, Greece

<sup>2</sup> Dipartimento di Informatica, Università di L'Aquila  
Via Vetoio, Coppito 67100, L'Aquila, Italy

**Abstract.** In this paper we present a new approximation algorithm for the *Minimum Energy Broadcast Routing* (MEBR) problem in ad hoc wireless networks that has exponentially better approximation factor than the well-known Minimum Spanning Tree (MST) heuristic. Namely, for any instance where a minimum spanning tree of the set of stations is guaranteed to cost at most  $\rho$  times the cost of an optimal solution for MEBR, we prove that our algorithm achieves an approximation ratio bounded by  $2 \ln \rho - 2 \ln 2 + 2$ . This result is particularly relevant for its consequences on Euclidean instances where we significantly improve previous results.

## 1 Introduction

Over the last years the usage of *wireless networks* has seen a huge increase mostly because of the recent drop in equipment prices and due to the features provided by the new technologies. In particular, considerable attention has been devoted to the so-called *ad hoc* wireless networks, due to their potential applications in emergency disaster relief, battlefield, etc. [16,25]. Ad hoc networks do not require any fixed infrastructure. The network is simply a collection of homogeneous radio stations equipped with omnidirectional antennas for sending and receiving signals. Communication occurs by assigning to each station a transmitting power. In the most common power attenuation model [22,24], the signal power  $P_s$  of a station  $s$  decreases as a function of the distance in such a way that at any station  $t$  at distance  $dist(s, t)$  it is received with a power  $\frac{P_s}{dist(s, t)^\alpha}$ , where  $\alpha \geq 1$  is a constant called the *distance-power gradient*. While for practical purposes the constant  $\alpha$  is usually assumed to be between 2 and 5, from a theoretical point of view general values of  $\alpha$  are also of interest. The signal is correctly received by  $t$  if  $\frac{P_s}{dist(s, t)^\alpha} \geq \beta$ , where  $\beta \in \mathbb{R}^+$  is the *transmission quality threshold*.

---

\* This work was partially supported by the EU COST Action 293 “Graphs and Algorithms in Communication Networks” (GRAAL) and by the EU IST FET Integrated Project 015964 AEOLUS.



Therefore, if  $s$  correctly transmits within a given maximum distance  $r(s)$ , called the range of  $s$ , the transmission power  $P_s$  of  $s$  is at least  $\beta \cdot r(s)^\alpha$ . Usually, the transmission quality threshold  $\beta$  is normalized to 1. We remark that, due to the nonlinear power attenuation, multi-hop transmission of messages through intermediate stations may result in energy saving.

A naturally arising issue in ad hoc wireless networks is that of supporting communication patterns that are typical in traditional networks, such as broadcasting (one-to-all), multicasting (one-to-many), and gossiping (all-to-all), with a minimum total energy consumption. This problem is generally called *Minimum Energy Routing* (MER) and defines different optimization subproblems according to the connectivity requirements (see for instance [4,5,7,8,11,12,15,17,19,20] for related results). In this paper, we are interested in the broadcast communication from a given source node.

Formally, given a set of stations  $S$ , let  $G(S)$  be the complete weighted graph whose nodes are the stations of  $S$  and in which the weight  $w(x, y)$  of each edge  $\{x, y\}$  is the power consumption needed for a correct communication between  $x$  and  $y$ . A power assignment for  $S$  is a function  $p : S \rightarrow \mathbb{R}^+$  assigning a transmission power  $p(x)$  to every station in  $S$ . A power assignment  $p$  for  $S$  yields a directed communication graph  $G^p = (S, A)$  such that, for each  $(x, y) \in S^2$ , the directed edge  $(x, y)$  belongs to  $A$  if and only if  $p(x) \geq w(x, y)$ , that is if  $x$  can correctly transmit to  $y$ . In this case  $y$  is also said to fall within the transmission range of  $x$ . The total cost of a power assignment is

$$\text{cost}(p) = \sum_{x \in S} p(x).$$

The MEBR problem for a given source  $s \in S$  consists in finding a power assignment  $p$  of minimum cost  $m^*(S, s)$  such that  $G^p$  contains a directed spanning tree rooted at  $s$  (and directed towards the leaves).

In general, the problem is unlikely to have polynomial-time approximation algorithms with approximation ratio  $o(\ln n)$  [9], where  $n$  is the number of stations. Logarithmic (in the number of stations) approximation algorithms have been presented in [4,6,7].

An important case of practical interest is when stations lie in a  $d$ -dimensional Euclidean space. Then, given a constant  $\alpha \geq 1$ , the power consumption needed for a correct communication between  $x$  and  $y$  is  $\text{dist}(x, y)^\alpha$ , where  $\text{dist}(x, y)$  is the Euclidean distance between the locations of  $x$  and  $y$ . The problem has been proved to be NP-hard for  $\alpha > 1$  and  $d > 1$ , while it is solvable in polynomial time for  $\alpha = 1$  or  $d = 1$  [6,9,3].

Several attempts in the literature were made to find good approximation algorithms for Euclidean cases. One fundamental algorithm to provide an approximate solution of the MEBR problem is the MST heuristic [24]. It is based on the idea of tuning ranges so as to include a minimum spanning tree of the cost graph  $G(S)$ . More precisely, denote by  $T(S)$  a minimum spanning tree of  $G(S)$ . The MST heuristic considers  $T(S)$  rooted at the source station  $s$ , directs the edges of  $T(S)$  towards the leaves, and sets the power  $p(x)$  of every internal station  $x$  of  $T(S)$  with  $k > 0$  children  $x_1, \dots, x_k$  in such a way

that  $p(x) = \max_{i=1,\dots,k} w(x, x_i)$ . In other words,  $p$  is the power assignment of minimum cost inducing the directed tree derived from  $T(S)$  and is such that  $\text{cost}(p) \leq c(T(S))$ , where  $c(T(S))$  denotes the total cost of the edges in  $T(S)$ . Therefore, the approximation ratio of the heuristic is bounded by the ratio between the cost of a minimum spanning tree of  $G(S)$  and the optimal cost  $m^*(S, s)$ .

The performance of the MST heuristic has been investigated by several authors [1,6,9,13,18,21,23]. The analysis in all the above papers focuses on the case  $\alpha \geq d$  (MST has unbounded approximation ratio when  $\alpha < d$ ) and is based on elegant geometric arguments. The best known approximation ratios are 6 for  $d = 2$  [1], 18.8 for  $d = 3$  [21] and  $3^d - 1$  for every  $d > 3$  [13]. Moreover, in [9,23], a lower bound on the approximation ratio of the MST heuristic has been proven, upper-bounding it by the  $d$ -dimensional kissing number  $n_d$ , i.e., the maximum number of  $d$ -dimensional unit spheres that touch a unit sphere but are mutually non-overlapping (but possibly touch each other). This number is 6 for  $d = 2$  (and, hence, the upper bound of [1] is tight), 12 for  $d = 3$  and, in general,  $n_d = 2^{cd(1+o(1))}$  with  $0.2075 \leq c \leq 0.401$  for large  $d$  [10]. Despite the considerable research effort in the area during the past years, no algorithm has been theoretically shown so far to outperform the MST heuristic in the Euclidean case, and the improvement of the corresponding ratios is a long standing open question.

Several other heuristics have been shown to perform better than MST in practice, at least for 2-dimensional instances (e.g., see [2,3,8,14,24]). The most famous among them is probably algorithm BIP (broadcast incremental power [24]). Starting from the source, it builds a tree in steps as follows: at each step, it includes the edge to an uncovered station that requires the minimum increase of power. BIP has been shown to be at least as good as MST while the best known lower bound on its approximation ratio is 4.33 [23]. All other heuristics that seem to work well in practice are either very complicated to analyze or have high lower bounds in terms of their approximation ratio.

In this paper we present a new approximation algorithm for the MEBR problem. For any instance of the problem where the minimum spanning tree of the cost graph  $G(S)$  is guaranteed to cost at most  $\rho$  times the cost of an optimal solution for MEBR, our algorithm achieves an approximation ratio bounded by  $2 \ln \rho - 2 \ln 2 + 2$  if  $\rho > 2$  and bounded by  $\rho$  if  $\rho \leq 2$ , which exponentially improves upon the MST heuristic. Surprisingly, our algorithm and analysis does not make use of any geometric arguments and still our results significantly improve the previously best known approximation factor for Euclidean instances of the problem. The corresponding approximation ratio is reduced (when  $\alpha \geq d$ ) from 6 [1] to 4.2 for  $d = 2$ , from 18.8 [21] to 6.49 for  $d = 3$  and in general from  $3^d - 1$  [13] to  $2.20d + 0.61$  for  $d > 3$ . In the 2-dimensional case, the achieved approximation is even less than the lower bound on the approximation ratio of the BIP heuristic. In arbitrary (i.e., non-Euclidean) cost graphs, it is not difficult to see that the cost of the minimum spanning tree is at most  $n - 1$  times the cost of an optimal solution for MEBR; hence, our algorithm also slightly improves the logarithmic

approximations of [4,6,7]. We also prove that our analysis is tight by showing that there are instances in which the ratio among the cost of the solution returned by the algorithm and the cost of the optimal solution is arbitrarily close to  $2 \ln \rho - 2 \ln 2 + 2$ .

The rest of the paper is organized as follows. In Section 2, we describe the new approximation algorithm and, in Section 3, we prove its correctness. In Section 4, we show that our analysis is tight, and, finally, in Section 5, we give some conclusive remarks.

## 2 The Approximation Algorithm

In this section, we describe our approximation algorithm. We begin with some necessary definitions.

Given a power assignment  $p$  and a given station  $x \in S$ , let  $E(p, x) = \{\{x, y\} \mid w(x, y) \leq p(x)\}$  be the set of the undirected edges induced by  $p$  at  $x$ , and  $E(p) = \bigcup_{x \in S} E(p, x)$  the set of all the undirected edges induced by  $p$ .

In the following, for every subset of undirected edges  $F \subseteq E$  of a weighted graph  $G = (V, E)$ , we will denote as  $c(F)$  the overall cost of the edges in  $F$ , that is the total sum of their weights. For the sake of simplicity, we will identify trees with their corresponding sets of edges. Given an undirected tree  $T$  and two nodes  $u$  and  $v$  of  $T$ ,  $u, v \in V$ , let  $P(T, u, v) \subseteq T$  be the subset of the edges in the unique path from  $u$  to  $v$  in  $T$ .

A *swap set* for a spanning tree  $T$  of an undirected graph  $G = (V, E)$  and a set of edges  $F$  with endpoints in  $V$  is any subset  $F'$  of edges that must be removed from the multigraph  $T \cup F$  in order to eliminate cycles and so that  $T \cup F \setminus F'$  is a spanning tree of  $G$ .

We are now ready to describe the algorithm. Before going through the details, let us describe the basic underlying idea. Starting from a spanning tree  $T(S)$  of  $G(S)$ , if the cost of  $T$  is significantly higher than the one of an optimal solution for performing broadcasting from a given source  $s \in S$ , then there must exist a cost efficient *contraction* of  $T$ . Namely, it must be possible to set the transmission power  $p(x)$  of at least one station  $x$  in such a way that  $p(x)$  is much lower than the cost of the swap set  $A(p, x)$  for  $T(S)$  and  $E(p, x)$ . The algorithm then repeatedly chooses at each step  $p(x)$  in such a way that, starting from the current spanning tree,  $c(A(p, x))/p(x)$  is maximized. The final tree will be such that, considering the correct orientation of the edges according to the final assignment  $p$ , some edges will be in the reverse direction, i.e., from the leaves towards the source  $s$ . However, the transmission powers can then be properly set with low additional cost in order to obtain the right orientation from  $s$  towards the other stations.

At a given intermediate step of the algorithm in which  $p$  and  $T$  are the current power assignment and maintained tree, respectively, consider a contraction at a given station  $x$  consisting of setting the transmission power of  $x$  to  $p'(x)$ , and let  $p'$  be the resulting power assignment. Then, a maximum cost swap set  $A(p', x)$

---

<sup>1</sup> To the purposes of the algorithm, we need to maintain in  $T \cup F$  at most two copies of the same edge with different weights.

to be accounted to the contraction can be trivially determined by letting  $A(p', x)$  contain the edges that are removed when determining a minimum spanning tree in the multigraph  $T \cup E(p', x)$  with the cost of all the edges in  $E(p', x)$  set equal to 0. Let the ratio  $\frac{c(A(p', x))}{p'(x)}$  be the cost-efficiency of the contraction.

The algorithm then performs the following steps:

- Set the transmission power  $p(x)$  of every station in  $x \in S$  equal to 0.
- Let  $T = T(S)$  be a minimum spanning tree of  $G(S)$ .
- While there exists at least one contraction of cost-efficiency strictly greater than 2:
  - Perform a contraction of maximum cost-efficiency, and let  $p'(x)$  be the corresponding increased power at a given station  $x$ , and  $p'$  be the resulting power assignment.
  - Set the weight of all the edges in  $E(p', x)$  equal to 0.
  - Let  $T' = T \cup E(p', x) \setminus A(p', x)$ .
  - Set  $T = T'$  and  $p = p'$ .
- Orient all the edges of  $T$  from the source  $s$  toward all the other stations.
- Return the transmission power assignment  $p$  that induces such a set of oriented edges.

Notice that if  $\rho \leq 2$  the algorithm performs no contraction step and it returns the initial minimum spanning tree  $T(S)$ ; thus, it guarantees the same approximation ratio  $\rho$  of the MST heuristic. Therefore, in the following of the paper we will assume that  $\rho > 2$ .

### 3 Correctness of the Algorithm

Clearly, the algorithm has a running time polynomial in the size of the input instance since, at each step, the power of some station increases while the power of the remaining nodes does not decrease. We thus now focus on the proof of the achieved approximation ratio.

We first give two lemmas which are very useful in order to show the existence of good contractions. Due to lack of space, the corresponding proofs have been omitted from this extended abstract. They will appear in the final version of the paper.

**Lemma 1.** *Given two rooted spanning trees  $T_1$  and  $T_2$  over the same set of nodes  $V$ , there exists a one-to-one mapping  $f : T_1 \rightarrow T_2$ , called the swap mapping, such that, if  $v_1, \dots, v_k$  are all the children of a same parent node  $u$  in  $T_1$ , then the set  $\{f(\{u, v_1\}), \dots, f(\{u, v_k\})\}$  of the edges assigned to  $\{u, v_1\}, \dots, \{u, v_k\}$  by  $f$  is a swap set for  $T_2$  and  $\{\{u, v_1\}, \dots, \{u, v_k\}\}$ .*

**Lemma 2.** *Given any tree  $T$ , and  $k$  edges  $\{u, v_1\}, \dots, \{u, v_k\}$  not necessarily belonging to  $T$ , if  $\{\{w_1, y_1\}, \dots, \{w_k, y_k\}\}$  is a swap set for  $T$  and  $\{\{u, v_1\}, \dots, \{u, v_k\}\}$ , then  $\{\{w_1, y_1\}, \dots, \{w_k, y_k\}\}$  is the subset of a swap set for  $T$  and  $\{\{u, v_1\}, \dots, \{u, v_k\}, \{u, z_1\}, \dots, \{u, z_l\}\}$ , for every set of  $l$  newly added edges  $\{\{u, z_1\}, \dots, \{u, z_l\}\}$ .*

We are now ready to prove the fundamental property that our algorithm exploits.

**Lemma 3.** *Let  $T$  be any spanning tree for  $G(S)$  with an arbitrary weighting of the edges, and let  $\gamma = c(T)/m^*(S, s)$  be the ratio among the cost of  $T$  and the one of an optimal transmission power assignment  $p^*$ . Then there exists a contraction of  $T$  of cost-efficiency  $\gamma$ .*

*Proof.* Consider a spanning tree  $T^*$  of  $G^{p^*}$ , and let  $f$  be the swap mapping for  $T^*$  and  $T$  derived from Lemma 1 considering  $T^*$  rooted at  $s$ .

Then, by Lemma 1,  $f$  assigns to all the descending edges  $D(T^*, x)$  in  $T^*$  of every station  $x \in S$  a subset of edges  $SS(x) \subseteq T$  forming a swap set for  $T$  and  $D(T^*, x)$ . All such subsets  $SS(x)$  form a partition of  $T$ , and since  $\frac{c(T)}{m^*(S, s)} = \frac{\sum_{x \in S} c(SS(x))}{\sum_{x \in S} p^*(x)} = \gamma$ , there must exist at least one station  $x$  such that  $\frac{c(SS(x))}{p^*(x)} \geq \gamma$ .

Since  $D(T^*, x) \subseteq E(p^*, x)$ , by Lemma 2,  $SS(x) \subseteq A(p^*, x)$ , where  $A(p^*, x)$  is a swap set for  $T$  and  $E(p^*, x)$ . Therefore, there exists a contraction of  $T$  of cost-efficiency  $c(A(p^*, x))/p^*(x) \geq c(SS(x))/p^*(x) = \gamma$ .  $\square$

By exploiting Lemma 3, we can prove the following upper bound on the approximation ratio of our algorithm.

**Theorem 1.** *Given an instance of MEBR consisting of a cost graph  $G(S)$  and a source station  $s \in S$ , the algorithm has approximation ratio  $2 \ln \rho - 2 \ln 2 + 2$ , where  $\rho > 2$  is the ratio of the cost of the minimum spanning tree over  $G(S)$  over the cost of an optimal solution for the MEBR instance.*

*Proof.* Let  $T_0 = T(S)$  be the minimum spanning tree for  $G(S)$  computed at the beginning of the algorithm,  $T_1, \dots, T_k$  be the sequence of the trees constructed by the algorithm after the contraction steps, that for the sake of clarity we assume numbered from 0 to  $k - 1$ , and  $\gamma_i = c(T_i)/m^*(S, s)$  be the ratio among the cost of  $T_i$  and the one of an optimal transmission power assignment  $p^*$ .

By applying Lemma 3, since the algorithm always considers contractions of maximum cost-efficiency, at each step  $i = 0, 1, \dots, k - 1$  it performs a contraction having cost-efficiency at least  $\gamma_i$ .

If  $\gamma_0 \leq 2$ , the algorithm performs no contraction step and it returns the initial minimum spanning tree  $T_0$ ; thus, the achieved approximation ratio is  $\gamma_0 \leq 2 \ln \rho - 2 \ln 2 + 2$  as  $\rho > 2$ . In the remaining part of the proof we will assume  $\gamma_0 > 2$ .

Let  $x_i$  be the node involved in the contraction of step  $i$  and  $p_i$  be the resulting transmission power assigned to  $x_i$ . Let  $t_i = c(T_i) - c(T_{i+1})$  be the cost of the edges belonging to the original spanning tree  $T_0$  removed by the algorithm at step  $i$ , i.e. included in the maximum cost swap set  $A(p_i, x_i)$  (only such edges have non-zero weights and thus contribute to the cost of  $A(p_i, x_i)$ ).

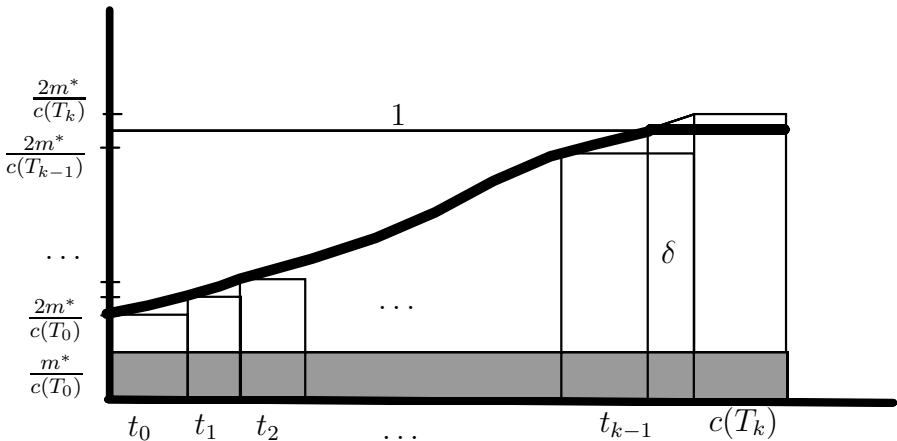
A power assignment inducing all the edges of  $T_k$  oriented from  $s$  towards all the other stations can be obtained by assigning to all the nodes a power assignment equal to the maximum weight of its outgoing edges in  $T_k$  with non-zero weight.

Moreover, given the power assignment  $p$  determined by the algorithm right at the end of the contraction steps, orienting all the corresponding edges of  $T_k$  with zero weight in the right direction requires at most doubling the cost of  $p$ . Indeed, consider a node  $x$  which is connected through edges of zero weight to  $\ell > 0$  children  $x_1, \dots, x_\ell$  in  $T_k$  (according to the orientation of  $T_k$  from the source node  $s$ ). Then, in order to make the power assignment induce these edges with direction from  $x$  to the children  $x_1, \dots, x_\ell$ , it suffices to increase the power of node  $x$  by at most  $\max_{i=1, \dots, \ell} p(x_i) \leq \sum_{i=1}^{\ell} p(x_i)$ . Hence, the final power assignment has overall cost upper-bounded by  $2 \sum_{i=0}^{k-1} p_i + c(T_k)$ .

Since by the definition of the algorithm the last contraction has cost-efficiency  $\gamma_{k-1} > 2$ , and afterwards no contraction of cost-efficiency greater than 2 exists, by Lemma 3 the cost  $c(T_k)$  of the final tree is at most  $2m^*(S, s)$ . Denoting by  $m$  the cost of the final power assignment returned by the algorithm we obtain

$$\begin{aligned} m &\leq 2 \sum_{i=0}^{k-1} \frac{t_i}{\gamma_i} + c(T_k) \\ &\leq 2 \sum_{i=0}^{k-2} \frac{t_i}{\gamma_i} + 2 \frac{t_{k-1} - \delta}{\gamma_{k-1}} + \delta + c(T_k) \\ &= 2 \sum_{i=0}^{k-2} \frac{t_i}{\gamma_i} + 2 \frac{t_{k-1} - \delta}{\gamma_{k-1}} + 2m^*(S, s), \end{aligned}$$

where  $\delta = 2m^*(S, s) - c(T_k)$ .



**Fig. 1.** The cost of the optimal solution (gray area); the cost of the MST heuristic solution (area of the rectangle of height 1); the cost of the solution returned by the algorithm (area below the bold line)

By recalling that  $\gamma_i = c(T_i)/m^*(S, s)$ , we finally have (see Figure [III](#))

$$\begin{aligned}
 m &\leq 2 \sum_{i=0}^{k-2} \frac{t_i}{\gamma_i} + 2 \frac{t_{k-1} - \delta}{\gamma_{k-1}} + 2m^*(S, s) \\
 &= 2m^*(S, s) \left( \sum_{i=0}^{k-2} \frac{t_i}{c(T_i)} + \frac{t_{k-1} - \delta}{c(T_{k-1})} + 1 \right) \\
 &\leq 2m^*(S, s) \left( \sum_{i=0}^{k-2} \int_0^{t_i} \frac{dt}{c(T_i) - t} + \int_0^{t_{k-1} - \delta} \frac{dt}{c(T_{k-1}) - t} + 1 \right) \\
 &= 2m^*(S, s) \left( \sum_{i=0}^{k-2} \int_{c(T_0) - c(T_i)}^{c(T_0) - c(T_{i+1})} \frac{dt}{c(T_0) - t} + \int_{c(T_0) - c(T_{k-1})}^{c(T_0)(1 - \frac{2}{\gamma_0})} \frac{dt}{c(T_0) - t} + 1 \right) \\
 &= 2m^*(S, s) \left( \int_0^{c(T_0)(1 - \frac{2}{\gamma_0})} \frac{dt}{c(T_0) - t} + 1 \right) \\
 &= m^*(S, s) (2 \ln \gamma_0 - 2 \ln 2 + 2).
 \end{aligned}$$

If the initial minimum spanning tree  $T_0 = T(S)$  guarantees a  $\rho$ -approximation of an optimal solution, the theorem follows by observing that  $\gamma_0 \leq \rho$ .  $\square$

## 4 A Matching Lower Bound

In this section we present a matching lower bound on the approximation ratio of our algorithm, i.e., we show that our analysis is tight.

**Theorem 2.** *For any  $\epsilon > 0$ , there exists an instance of MEBR consisting of a cost graph and a source station for which the solution returned by the algorithm has cost at least  $2 \ln \rho - 2 \ln 2 + 2 - \epsilon$  times the optimal cost, where  $\rho > 2$  is the ratio between the cost of the minimum spanning tree over the cost graph and the cost of an optimal solution for MEBR.*

*Proof.* In order to describe the considered instance, it is useful to first describe the *building block*  $Q_x$  with  $x > 2$ , depicted in Figure [2](#).

The set of nodes and edges of  $Q_x$  are  $V_x = \{v_x\} \cup \{u_{x,i} | i = 1, 2, \dots, \lceil x \rceil\}$ , and  $E_x = \{\{v_x, u_{x,i}\} | i = 1, 2, \dots, \lceil x \rceil\}$ , respectively. The weight of the edge  $\{v_x, u_{x, \lceil x \rceil}\}$  is equal to  $1 + x - \lceil x \rceil$ , and the weights of all the other edges are equal to 1. Notice that in  $Q_x$  there exists a contraction centered at node  $v_x$  having cost-efficiency equal to  $x$ .

We are now ready to describe the whole instance, whose minimum spanning tree is depicted in Figure [3](#). Let  $k$  be an integer parameter; the node set of the instance is obtained by sequencing  $k(\rho - 2)$  building blocks  $(Q_{2+\frac{1}{k}}, Q_{2+\frac{2}{k}}, \dots, Q_3, \dots, Q_\rho)$  in such a way that for two consecutive blocks  $Q_x$  and  $Q_y$ ,  $x > y$ , nodes  $u_{x,2}$  and  $u_{y,1}$  coincide. Moreover, in the instance there are other 3 nodes: the source  $s$ , node  $v_1$ , and node  $v_2$  which coincides with  $u_{2+\frac{1}{k},2}$ .

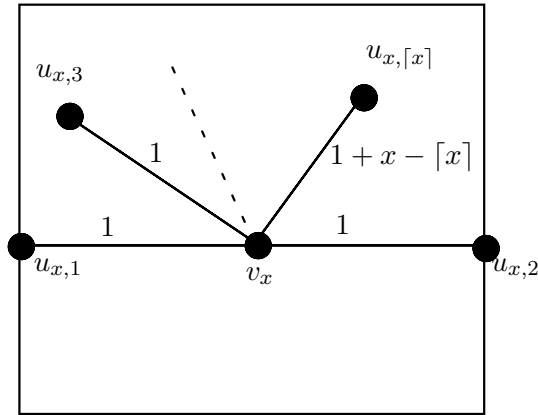


Fig. 2. The building block  $Q_x$

It remains to define the weights of the edges between the nodes. The weights of the edges connecting  $s$  to all the other nodes are equal to 1; moreover,  $w(v_1, v_2) = 1$ . The weights of the edges contained in the building blocks are properly scaled so that the sum of all the edges of each building block is equal to  $\frac{1}{k}$ . In particular, the weights of all the edges belonging to building block  $Q_x$  are divided by  $kx$ . For all the other pairs of nodes, we assume that the mutual power communication cost is very high.

Assume that the initial minimum spanning tree considered by the algorithm is the one depicted in Figure 3, whose cost is  $\rho$ .

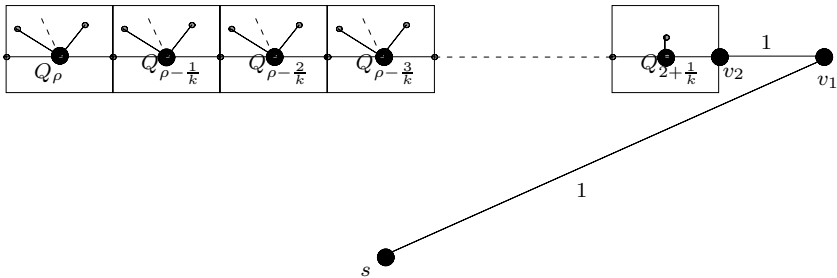


Fig. 3. A minimum spanning tree of the lower bound instance

At the initial step (step 0), the algorithm can arbitrarily choose among two equivalent contractions, i.e., having the same cost-efficiency; the first choice is the contraction centered at the source and having transmission power equal to 1, and the second choice is the contraction centered at  $v_\rho$  and having transmission power equal to  $\frac{1}{\rho}$ . Both contractions have a cost-efficiency equal to  $\rho$ , and we assume that the algorithm chooses the contraction centered at  $v_\rho$ .



Using the same arguments, we can assume that the algorithm proceeds by performing other  $k(\rho - 2) - 1$  steps of contractions (steps  $1, 2, \dots, k(\rho - 2) - 1$ ), choosing at step  $i$  the contraction centered at  $v_{\rho - \frac{i}{k}}$ , having transmission power equal to  $\frac{1}{k\rho - i}$  and cost-efficiency equal to  $\rho - \frac{i}{k}$ , instead of the equivalent contraction centered at  $s$ .

At this point, no contraction having cost-efficiency at least 2 exists any longer. Notice that the sum of the costs of the transmission powers set in the contractions is  $\sum_{i=2k+1}^{k\rho} \frac{1}{i} = H_{k\rho} - H_{2k}$ , where  $H_i = 1 + \frac{1}{2} + \dots + \frac{1}{i}$  is the harmonic number. In order to orient the edges of the final tree from the source towards the node, we have to globally double the cost of the transmission powers set in the contraction steps. In particular, we have to assign to  $v_2$  a transmission power equal to the one of  $v_{2+\frac{1}{k}}$  and to  $u_{x,2}$  a transmission power equal to the one of  $v_x$ , for  $x = 2 + \frac{2}{k}, 2 + \frac{3}{k}, \dots, \rho$ . Thus, the final cost of the solution returned by the algorithm has cost  $2H_{k\rho} - 2H_{2k} + 2$ , while the optimal solution has cost 1 and is obtained by assigning to the source node a transmission cost equal to 1. As it can be easily checked, letting  $k$  go to infinity, the approximation ratio tends to  $2 \ln \rho - 2 \ln 2 + 2$  from below. □

## 5 Conclusions

We have presented an approximation algorithm that exponentially outperforms the MST heuristic on any instance of the minimum energy broadcasting problem in ad hoc wireless networks. Our results are particularly relevant for their consequences on Euclidean instances where the achieved approximation ratio is linear in the number of dimension  $d$  instead of exponential. Therefore, the improvement becomes more and more significant as  $d$  increases. Some corresponding values are depicted in Figure 4.

| Number of dimensions        | 1       | 2     | 3         | 4       | 5        | 6        | 7         |
|-----------------------------|---------|-------|-----------|---------|----------|----------|-----------|
| MST heuristic approx. ratio | 2 [9,3] | 6 [1] | 18.8 [21] | 80 [13] | 242 [13] | 728 [13] | 2186 [13] |
| Our algorithm approx. ratio | 2       | 4.2   | 6.49      | 9.38    | 11.6     | 13.8     | 16        |

**Fig. 4.** Comparison between the approximation factors of our algorithm and the MST heuristic in Euclidean instances for an increasing number of dimensions

Several questions are left open. First of all, our analysis works on general instances, but further improvements might be possible for specific cases like the Euclidean ones. For such instances, it would be worth to determine exact results tightening the current gap between the lower and upper bounds on the approximation ratio. Another interesting issue is that of determining similar contraction strategies possibly leading to better approximate solutions. An important open question is also that of determining better approximation results of the MST heuristic on high-dimensional Euclidean instances. In particular, tightening the approximation ratio to the kissing number for any number of dimensions would

also decrease the approximation of our algorithms, although the improvement would be restricted only to a constant multiplicative factor.

## References

1. Ambühl, C.: An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1139–1150. Springer, Heidelberg (2005)
2. Athanassopoulos, S., Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: Experimental comparison of algorithms for energy-efficient multicasting in ad hoc networks. In: Nikolaidis, I., Barbeau, M., Kranakis, E. (eds.) ADHOC-NOW 2004. LNCS, vol. 3158, pp. 183–196. Springer, Heidelberg (2004)
3. Čagalj, M., Hubaux, J., Enz, C.: Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues. In: Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom '02), pp. 172–182. ACM Press, New York, NY, USA (2002)
4. Calinescu, G., Kapoor, S., Olshevsky, A., Zelikovsky, A.: Network lifetime and power assignment in ad-hoc wireless networks. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 114–126. Springer, Heidelberg (2003)
5. Calinescu, G., Mandoiu, I., Zelikovsky, A.: Symmetric connectivity with minimum power consumption in radio networks. In: Proceedings of the 2nd IFIP International Conference on Theoretical Computer Science, pp. 119–130. Kluwer, B.V., Dordrecht (2002)
6. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: New results for energy-efficient broadcasting in wireless networks. In: Bose, P., Morin, P. (eds.) ISAAC 2002. LNCS, vol. 2518, pp. 332–343. Springer, Heidelberg (2002)
7. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: Energy-efficient wireless network design. *Theory of Computing Systems* 39(5), 593–617 (2006)
8. Cartigny, J., Simplot, D., Stojmenovic, I.: Localized minimum-energy broadcasting in ad-hoc networks. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03), vol. 3, pp. 2210–2217. IEEE Computer Society Press, Los Alamitos (2003)
9. Clementi, A., Crescenzi, P., Penna, P., Rossi, G., Vocca, P.: On the complexity of computing minimum energy consumption broadcast subgraphs. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 121–131. Springer, Heidelberg (2001)
10. Conway, J.H., Sloane, N.J.A.: The kissing number problem and Bounds on kissing numbers (Ch. 2.1 and Ch. 13). In: *Sphere Packings, Lattices, and Groups*, 3rd edn. Springer, New York (1998)
11. Dong, Q., Banerjee, S., Adler, M., Misra, A.: Minimum energy reliable paths using unreliable wireless links. In: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05), pp. 449–459. ACM Press, New York (2005)
12. Doshi, S., Bhandare, S., Brown, T.X.: An on-demand minimum energy routing protocol for a wireless ad hoc network. *SIGMOBILE Mobile Computing and Communication Review* 6(3), 50–66 (2002)

13. Flammini, M., Klasing, R., Navarra, A., Perennes, S.: Improved approximation results for the minimum energy broadcasting problem. In: Proceedings of ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), pp. 85–91. ACM Press, New York (2004)
14. Flammini, M., Navarra, A., Perennes, S.: The real approximation factor of the MST heuristic for the minimum energy broadcasting. *ACM Journal of Experimental Algorithmics* 11(Article no. 2.10), 1–13 (2006)
15. Gilbert, E., Pollak, H.: Steiner minimal trees. *SIAM Journal on Applied Mathematics* 16, 1–29 (1968)
16. Hac, A.: *Wireless sensor network designs*. John Wiley & Sons, Ltd., West Sussex, England (2004)
17. Kirousis, L., Kranakis, E., Krizanc, D., Pelc, A.: Power consumption in packet radio networks. *Theoretical Computer Science* 243(1-2), 289–305 (2000)
18. Klasing, R., Navarra, A., Papadopoulos, A., Perennes, S.: Adaptive broadcast consumption (ABC), a new heuristic and new bounds for the minimum energy broadcast routing problem. In: Mitrou, N.M., Kontovasilis, K., Rouskas, G.N., Iliadis, I., Merakos, L. (eds.) *NETWORKING 2004*. LNCS, vol. 3042, pp. 866–877. Springer, Heidelberg (2004)
19. Li, X., Wan, P.: Constructing minimum energy mobile wireless networks. *SIGMOBILE Mobile Computing and Communication Review* 5(4), 55–67 (2001)
20. Liang, W.: Constructing minimum-energy broadcast trees in wireless ad hoc networks. In: Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '02), pp. 112–122. ACM Press, New York (2002)
21. Navarra, A.: 3-d minimum energy broadcasting. In: Flocchini, P., Gaşieniec, L. (eds.) *SIROCCO 2006*. LNCS, vol. 4056, pp. 240–252. Springer, Heidelberg (2006)
22. Rappaport, T.: *Wireless communications: principles and practice*. Prentice-Hall, Englewood Cliffs (1996)
23. Wan, P.-J., Calinescu, G., Li, X., Frieder, O.: Minimum energy broadcast routing in static ad hoc wireless networks. *Wireless Networks* 8(6), 607–617 (2002)
24. Wieselthier, J.-E., Nguyen, G.D., Ephremides, A.: On the construction of energy-efficient broadcast and multicast trees in wireless networks. In: Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00), pp. 585–594. IEEE Computer Society Press, Los Alamitos (2000)
25. Zhao, F., Guibas, L.: *Wireless sensor networks: an information processing approach*. Morgan Kaufmann, San Francisco (2004)

# Modular Algorithms for Heterogeneous Modal Logics

Lutz Schröder<sup>1,\*</sup> and Dirk Pattinson<sup>2</sup>

<sup>1</sup> DFKI-Lab Bremen and Department of Computer Science, Universität Bremen

<sup>2</sup> Department of Computing, Imperial College London

**Abstract.** State-based systems and modal logics for reasoning about them often heterogeneously combine a number of features such as non-determinism and probabilities. Here, we show that the combination of features can be reflected algorithmically and develop modular decision procedures for heterogeneous modal logics. The modularity is achieved by formalising the underlying state-based systems as multi-sorted coalgebras and associating both a logical and an algorithmic description to a number of basic building blocks. Our main result is that logics arising as combinations of these building blocks can be decided in polynomial space provided that this is the case for the components. By instantiating the general framework to concrete cases, we obtain *PSPACE* decision procedures for a wide variety of structurally different logics, describing e.g. Segala systems and games with uncertain information.

## 1 Introduction

Modal logics appear in computer science in a variety of contexts. They are the formalism of choice for reasoning about reactive systems and feature prominently in areas related to artificial intelligence such as knowledge representation and reasoning with uncertainty [7]. The semantics of modal logics typically involves a notion of state and transition, which can take a number of different forms. Transitions can be probabilistic or weighted, as in probabilistic modal logic [15,9] and graded modal logic [6,5], induced by joint actions of agents as in coalition logic [18], or non-monotonically conditioned as in conditional logic [3]. An attractive aspect of many of these logics is that they admit shallow models and are decidable in low complexity classes — in the absence of fixpoint operators typically *PSPACE* (e.g. [25,18,23]), i.e. the same as the standard modal logic *K* [2] and not dramatically worse than propositional logic.

Features like non-determinism, probabilistic choice or joint actions are often combined, leading to systems that incorporate more than one type of transition. Moreover, features can be combined in different ways: E.g. in the alternating model of probabilistic transition systems [8], states may have either non-deterministic or probabilistic transitions, whereas simple Segala systems [24] have a two-layered structure where non-deterministic transitions lead to probability distributions over successor states. Bartels et al. [1] present 12 different types of probabilistic transition systems that arise as such combinations of basic features.

Here, we introduce a simple calculus that formalises the combination of features and establish that combined logics inherit the pleasant properties of their building blocks,

---

\* Support by the DFG project HasCASL (KR 1191/7-2) is gratefully acknowledged.

in particular shallow models and decidability in *PSPACE*. Our results and algorithms are *generic* and use the same algorithmic template to realise decision procedures at the level of each individual feature. This is achieved by formalising the combined logics in a multi-sorted extension of *coalgebraic modal logic* [17] whose semantics is parametric in a set functor  $T$ ; models then appear as  $T$ -coalgebras. This pushes the generic *PSPACE*-decision procedure of [23], which works uniformly for such diverse logics as Hennessy-Milner logic, coalition logic, graded modal logic, and probabilistic modal logic, to the level of combined logics that integrate several features.

Formally, a feature consists of a set of modal operators together with a set of associated proof rules. On the semantic level, a structure for a feature is an endofunctor of type  $\mathbf{Set}^n \rightarrow \mathbf{Set}$ , where  $n$  is the arity of the feature (e.g. choice, fusion, and conditionality are binary features). The notion of *gluing* formalises specific ways of combining given features. Syntactically, gluings define multi-sorted modal logics. Semantically, gluings induce endofunctors  $T : \mathbf{Set}^n \rightarrow \mathbf{Set}^n$  such that  $T$ -coalgebras are models of the combined logic. The single sorted case  $n = 1$  is of special interest since it captures the standard models of combined systems, including e.g. the ones presented in [1], which equip multi-sorted logics with a single-sorted semantics.

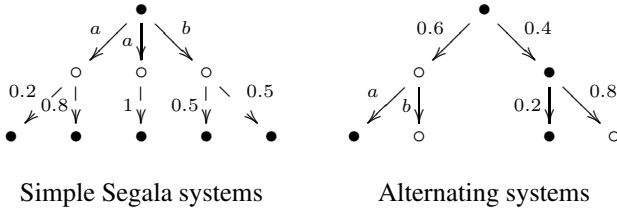
The central technical contribution of this work is the construction of a logically equivalent *flattening* of a given gluing, where flat gluings assign to each occurrence of a feature an individual sort in the semantics. Flat gluings are technically more tractable than general gluings. In particular, one can establish the shallow model property and a generic *PSPACE* algorithm for flat gluings. Together, these results imply *PSPACE* upper bounds for satisfiability w.r.t. general gluings, including the standard single-sorted semantics.

*Related Work.* Our work is closely related to the framework presented in [4,11], which focuses on completeness issues, with the main difference that our approach makes the multi-sorted nature of heterogeneous logics explicit by considering multi-sorted models. Our treatment of typed formulas resembles the use of *ingredients* in [11], but the multi-sorted semantics avoids the use of the next-operator of *loc.cit.* The main advantage of the new framework is that constructions such as cartesian product or disjoint union are no longer special cases and that the decision procedures of [22,23] generalise straightforwardly to the multi-sorted case. The multisorted approach to the complexity of composite modal logics complements transfer results obtained for the fusion of modal logics [26,10] in the sense that our framework is presently limited to logics axiomatised without nested modalities, but allows more flexible logic composition and covers also non-Kripke semantics.

## 2 Multisorted Modal Logics by Example

### 2.1 Logics for Probabilistic Systems

Segala systems [24] and alternating systems [8] both combine probabilistic transitions and non-determinism. In Segala systems, each system state can non-deterministically perform actions that lead to probability distributions over states. Contrastingly, alternating systems have two kinds of states engaging in purely probabilistic transitions and non-deterministic actions, respectively, that may end up in either kind of state.



It has been shown in [12, Theorem 8] that *probabilistic modal logic* over a set  $A$  of actions characterises states of image-finite Segala systems up to bisimilarity. This logic has two sorts  $n$  and  $u$  of non-deterministic and probabilistic (‘uncertain’) formulas, respectively, and two families of modal operators

$$\Box_a : u \rightarrow n \quad (a \in A) \quad \text{and} \quad L_p : n \rightarrow u \quad (p \in [0, 1] \cap \mathbb{Q}),$$

where  $L_p$  reads ‘with probability at least  $p$ ’. The sets  $\mathcal{L}_n$  and  $\mathcal{L}_u$  of non-deterministic and probabilistic formulas, respectively, are thus defined by the grammar

$$\begin{aligned} \mathcal{L}_n \ni \phi &::= \top \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \Box_a\psi & (\psi \in \mathcal{L}_u, a \in A) \\ \mathcal{L}_u \ni \psi &::= \top \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid L_p\phi & (\phi \in \mathcal{L}_n, p \in [0, 1] \cap \mathbb{Q}). \end{aligned}$$

Alternating systems, on the other hand, can be captured by a logic comprising three sorts  $n$ ,  $u$ , and  $o$  of non-deterministic, probabilistic, and alternating formulas, respectively, and modal operators

$$+ : u, n \rightarrow o \quad L_p : o \rightarrow u \quad \Box_a : o \rightarrow n$$

inducing the obvious three-sorted grammar. The binary modal operator  $+$  implements the choice between probabilistic and non-deterministic transitions, being essentially a case statement:  $\phi + \psi$  demands that  $\phi$  holds if the present state is probabilistic whereas  $\psi$  holds if present state is non-deterministic.

### 2.2 Fusion of Modal Logics

Both logics described above wire up the component logics in a restricted way, by imposing layering and choice, respectively. The unrestricted combination of logics  $\mathcal{L}_a$  and  $\mathcal{L}_b$  featuring modal operators  $\Box$  and  $\heartsuit$ , respectively, can be modelled by a logic with sorts  $a, b, f$  and four modal operators with associated source and target sorts

$$[\pi_1] : a \rightarrow f \quad [\pi_2] : b \rightarrow f \quad \Box : f \rightarrow a \quad \heartsuit : f \rightarrow b.$$

The  $[\pi_i]$  are postulated to commute with all boolean connectives. The well-known *fusion*  $\mathcal{L}_a \otimes \mathcal{L}_b$  (cf. e.g. [13]) disjointly combines the axioms and modalities of  $\mathcal{L}_a$  and  $\mathcal{L}_b$ . One can translate back and forth between the fusion and formulas of sort  $f$ , taking e.g. the operator  $\Box$  of the fusion to the composite operator  $[\pi_1]\Box$ . Thus, fusion is an instance of the multi-sorted combination of modal logics.

As fusion does not impose any well-typedness constraints on formulas, it can be regarded as the maximally permissive way of combining two modal logics. However, as shown by the previous example, formulas of the fusion do not in general have an interpretation over the intended type of systems, so that it is for many purposes preferable to work with the more restrictive well-typed combinations considered here.

### 2.3 Conditional Logic

The standard conditional logic  $CK$  [3] has a binary modal operator  $\Rightarrow$ , where  $\phi \Rightarrow \psi$  is read as a non-monotonic conditional. The right hand argument of  $\Rightarrow$  behaves essentially like the normal modal logic  $K$ , in particular obeys the usual  $K$ -axiom when the left argument is fixed. Indeed we can embed  $CK$  into a two-sorted *extended conditional logic* with sorts  $c, k$  and modal operators

$$\overset{\bullet}{\Rightarrow} : c, k \rightarrow c \quad \Box : c \rightarrow k$$

by translating  $\alpha \Rightarrow \beta$  to  $\alpha \overset{\bullet}{\Rightarrow} \Box\beta$ . Here,  $\overset{\bullet}{\Rightarrow}$  represents a rudimentary conditional, and  $\Box$  is the standard box modality of  $K$ . This shows how a given complex logic can be broken down into simpler building blocks.

## 3 Compositional Syntax of Multisorted Modal Logic

For our purposes, it is convenient to present the syntax of multi-sorted modal logic in a way that provides explicitly for a decomposition into building blocks. The building blocks, which we call *features*, are collections of (possibly polyadic) modal operators and associated proof rules that capture specific properties of a logic, such as the ability to describe choice, non-determinism, or uncertainty.

**Definition 1.** An  $n$ -ary *feature* is a pair  $F = (A, \mathcal{R})$  consisting of a set  $A$  of modal operators  $L$  with profiles  $L : i_1, \dots, i_k \rightarrow *$ , where  $1 \leq i_1, \dots, i_k \leq n$  are formal argument sorts and  $*$  is a formal target sort, and a set  $\mathcal{R}$  of *one-step rules* of the form  $(\phi_1; \dots; \phi_n)/\psi$ , where for  $i = 1, \dots, n$ ,  $\phi_i$  is a propositional formula over a set  $V_i$  of propositional variables, and  $\psi$  is a disjunctive clause over atoms of the form  $L(a_1, \dots, a_k)$  with  $L : i_1, \dots, i_k \rightarrow *$  in  $A$  and  $a_j \in V_{i_j}$ ,  $j = 1, \dots, k$ .

Note that the rule format disallows nested modalities in the conclusion, so that rules describe the one-step behaviour of a system. As in the single sorted case [22], this format always suffices to completely axiomatise the features of interest, as long as no global conditions (such as transitivity) are imposed on the coalgebraic models.

**Example 2.** We describe the features implicit in the Examples of Sect. 2. Figure 11 shows the associated proof rules, already in a special format needed in Sect. 5. The rules for non-determinism and uncertainty are taken from [23]; the others are obtained by the same principles. The sum expression in the uncertainty rule refers to the (propositionally expressible) arithmetic of characteristic functions [23].

**Non-Determinism:** Given a set  $A$  of actions, the unary feature  $N_A$  has modal operators  $\Box_a : 1 \rightarrow *$  for  $a \in A$ . We write  $K$  instead of  $N_A$  if  $A$  is a singleton.

**Uncertainty:** The unary feature  $U$  has modal operators  $L_p : 1 \rightarrow *$  for  $p \in [0, 1] \cap \mathbb{Q}$ .

**Choice:** The binary feature  $S$  has a single modal operator  $+$  :  $1, 2 \rightarrow *$ .

**Fusion:** The binary feature  $P$  has two modal operators  $[\pi_i] : i \rightarrow *$ ,  $i = 1, 2$ .

**Conditionality:** The binary feature  $C$  has a binary modal operator  $\overset{\bullet}{\Rightarrow} : 1, 2 \rightarrow *$ .

|                         |   |
|-------------------------|---|
| <b>Nondeterminism :</b> | $\frac{\bigwedge_{j=1}^n \alpha_j \rightarrow \beta}{\bigwedge_{j=1}^n \Box_a \alpha_j \rightarrow \Box_a \beta} \quad (n \geq 0, a \in A)$   |
| <b>Uncertainty :</b>    | $\frac{\sum_{j=1}^n r_j \alpha_j \geq k}{\bigvee_{j=1}^n \text{sgn}(r_j) L_{p_j} \alpha_j} \quad \left( \begin{array}{l} n \geq 1, r_j \in \mathbb{Z} - \{0\}, k \in \mathbb{Z} \\ \sum_{j=1}^n r_j p_j \begin{cases} < k & \text{if } \forall j. r_j < 0 \\ \leq k & \text{otherwise} \end{cases} \end{array} \right)$ |
| <b>Choice :</b>         | $\frac{(\bigwedge_{j=1}^m \alpha_j \rightarrow \bigvee_{k=1}^n \beta_k) : 1 \quad (\bigwedge_{j=1}^m \gamma_j \rightarrow \bigvee_{k=1}^n \delta_k) : 2}{\bigwedge_{j=1}^m (\alpha_j + \gamma_j) \rightarrow \bigvee_{k=1}^n (\beta_k + \delta_k)} \quad (m, n \geq 0)$   |
| <b>Fusion :</b>         | $\frac{(\bigwedge_{j=1}^m \alpha_j \rightarrow \bigvee_{k=1}^n \beta_k) : i}{\bigwedge_{j=1}^m [\pi_i] \alpha_j \rightarrow \bigvee_{k=1}^n [\pi_i] \beta_k} \quad (i = 1, 2; m, n \geq 0)$   |
| <b>Conditionality :</b> | $\frac{(\bigwedge_{j=1}^m \alpha_j \rightarrow \bigvee_{k=1}^n \beta_k) : 2}{\bigwedge_{j=1}^m (\gamma \dot{\Rightarrow} \alpha_j) \rightarrow \bigvee_{k=1}^n (\gamma \dot{\Rightarrow} \beta_k)} \quad (m, n \geq 0)$   |

**Fig. 1.** Proof rules for the features of Example 2

The examples from Sect. 2 demonstrate that features can be combined in different ways. This is formalised by the notion of gluing.

**Definition 3.** Let  $\Phi$  be a set of features, and let  $\mathcal{S}$  be a set of sorts. *Feature expressions*  $t$  are terms over the set  $\mathcal{S}$  of variables where the features appear as function symbols, i.e.

$$t ::= a \mid F(t_1, \dots, t_n) \quad a \in \mathcal{S}, F \in \Phi \text{ } n\text{-ary.}$$

A *gluing* of  $\Phi$  over  $\mathcal{S}$  is a family  $G = (t_a)_{a \in \mathcal{S}}$  of feature expressions, denoted by  $(a_1 \rightarrow t_{a_1}, \dots, a_n \rightarrow t_{a_n})$  for  $\mathcal{S} = \{a_1, \dots, a_n\}$ ; in this case we also write  $a_i \rightarrow t_{a_i} \in G$ .

A gluing  $G = (t_a)_{a \in \mathcal{S}}$  induces a multi-sorted modal logic, as follows. The set  $\text{Types}(G)$  of *G-types* consists of the proper subterms of the  $t_a$ , where the sorts  $a \in \mathcal{S}$  are called *base types* and the expressions  $t \in \text{Types}(G) \setminus \mathcal{S}$  are the *composite types*. (Types are related to the *ingredients* of [11].) We call a gluing *flat* if  $\mathcal{S} = \text{Types}(G)$ , i.e. there are no composite types, which is the case if every term  $t_a$  is of the form  $F(a_1, \dots, a_n)$ . Typed *G-formulas*  $\phi : s, s \in \text{Types}(G)$ , are inductively generated by closure under boolean operators  $\perp, \neg, \wedge$  at each type (with further boolean operators defined in the standard way) and by the following typing rules for composite types (left) and base types (right)

$$\frac{\phi_1 : s_1, \dots, \phi_n : s_n}{L(\phi_{i_1}, \dots, \phi_{i_n}) : F(s_1, \dots, s_n)} \quad \frac{\phi_1 : s_1, \dots, \phi_n : s_n}{L(\phi_{i_1}, \dots, \phi_{i_n}) : a'}$$

where the left hand rule has side condition  $F(s_1, \dots, s_n) \in \text{Types}(G)$  and the right hand rule has side condition  $a \rightarrow F(s_1, \dots, s_n) \in G$ , and in both cases  $L : i_1, \dots, i_n \rightarrow *$  in  $F$ . We write  $\mathcal{F}_s(G)$  for the set of *G-formulas* of type  $s$  and denote the family  $(\mathcal{F}_s(G))_{s \in \text{Types}(G)}$  by  $\mathcal{F}(G)$ .

Similarly, the proof system induced by  $G$  is described in terms of a  $\text{Types}(G)$ -indexed family of derivability predicates  $\vdash_s \subseteq \mathcal{F}_s(G)$  defined inductively by closure



under propositional reasoning at each type and the deduction rules for composite types (left) and base types (right), distinguished only by the type discipline,

$$\frac{\vdash_{s_1} \phi_1 \sigma, \dots, \vdash_{s_n} \phi_n \sigma}{\vdash_{F(s_1, \dots, s_n)} \psi \sigma} \quad \frac{\vdash_{s_1} \phi_1 \sigma \dots \vdash_{s_n} \phi_n \sigma}{\vdash_a \psi \sigma}$$

where  $F(s_1, \dots, s_n) \in \text{Types}(\mathbb{G})$  in the left hand rule,  $a \rightarrow F(s_1, \dots, s_n) \in \mathbb{G}$  in the right hand rule, and in both cases,  $(\phi_1; \dots; \phi_n)/\psi$  is a rule of  $F$  and  $\sigma$  is a substitution mapping variables  $a \in V_i$  to formulas  $\sigma(a) : s_i$ .

A given logic can be syntactically generated by different gluings, typically including both flat and non-flat ones, determining different classes of semantic structures (cf. Sect. 4). The core of this work is the proof of logical equivalence for the respective semantics. Flat gluings are technically more tractable, while logics occurring in the literature, including the ones described in Sect. 2 are typically non-flat.

**Example 4.** From the features  $S$ ,  $U$ , and  $N_A$  (Example 2), we can form gluings

$$G_1 \equiv (a \rightarrow S(U(a), N_A(a))) \text{ and } G_2 \equiv (a \rightarrow S(u, n), u \rightarrow U(a), n \rightarrow N_A(a)).$$

Here,  $G_1$  has types  $a, N_A(a), U(a)$ , whereas  $G_2$  is flat with types  $a, n, u$ . Modulo identifications  $N_A(a) = n$  and  $U(a) = u$ , both gluings give rise to the (typed) formulas describing alternating systems (Sect. 2.1). The remaining example logics from Sect. 2 are captured by the following gluings (where we omit the obvious flat versions):

**Probabilistic modal logic of Segala Systems:**  $s \rightarrow N_A(U(s))$ .

**Fusion:** The fusion of logics  $\mathcal{L}_a$  and  $\mathcal{L}_b$  as in Sect. 2.2 regarded as features, is  $f \rightarrow P(\mathcal{L}_a(f), \mathcal{L}_b(f))$ .

**Extended conditional logic:**  $c \rightarrow C(c, K(c))$ . Note in particular that in the induced proof system, we can derive the standard rule

$$(RCK) \frac{\bigwedge_{i=1}^n \alpha_i \rightarrow \beta}{\bigwedge_{i=1}^n (\gamma \Rightarrow \alpha_i) \rightarrow (\gamma \Rightarrow \beta)} \quad (n \geq 0)$$

of the conditional logic  $CK$  [3], where  $\gamma \Rightarrow \alpha$  abbreviates  $\gamma \overset{\bullet}{\Rightarrow} \Box \alpha$ .

## 4 Multi-sorted vs. Single-Sorted Coalgebraic Semantics

We now generalise the coalgebraic interpretation of modal logic, introduced in [17], to the multi-sorted case. Crucially, we interpret multi-sorted logics over multi-sorted coalgebras. The parametricity over signature functors for coalgebras is the key feature of our framework that allows for uniform results that can be instantiated to a large number of structurally different systems and logics. We recall some basic notions of multi-sorted coalgebra (cf. e.g. [16]), generalising the single-sorted setting [19]:

**Definition 5.** We write  $\mathbf{Set}$  for the category of sets and functions. Let  $\mathbf{Set}^S$  denote the category of  $S$ -sorted sets and  $S$ -sorted functions, with objects being families  $X = (X_a)_{a \in S}$  (or just  $(X_a)$ ) of sets  $X_a$ , and morphisms  $f : (X_a) \rightarrow (Y_a)$  being families

$f = (f_a)_{a \in \mathcal{S}}$  of maps  $f_a : X_a \rightarrow Y_a$ . We write  $\mathbf{Set}^n$  for  $\mathbf{Set}^{\{1, \dots, n\}}$ . A functor  $T : \mathbf{Set}^{\mathcal{S}} \rightarrow \mathbf{Set}^{\mathcal{S}}$  may be regarded a family  $T = (T_a)_{a \in \mathcal{S}}$  of functors  $T_a : \mathbf{Set}^{\mathcal{S}} \rightarrow \mathbf{Set}$ . A  $T$ -coalgebra  $A = (X, \xi)$  is a pair  $(X, \xi)$  where  $X$  is an  $\mathcal{S}$ -sorted set and  $\xi = (\xi_a) : X \rightarrow TX$  is an  $\mathcal{S}$ -sorted function (i.e.  $\xi_a : X_a \rightarrow T_a X$ ) called the *transition function*. A *morphism* between  $T$ -coalgebras  $(X, \xi)$  and  $(Y, \zeta)$  is an  $\mathcal{S}$ -sorted function  $f : X \rightarrow Y$  such that  $(Tf)\xi = \zeta f$  in  $\mathbf{Set}^{\mathcal{S}}$ .

We view coalgebras as generalised transition systems: the transition function maps states to structured sets of observations and successor states, the latter taken from the available sorts as specified by  $T$ .

The interpretation of modal operators is based on predicate liftings [17, 20]; in the multi-sorted setting, this takes the following shape.

**Definition 6.** A *predicate lifting*  $\lambda$  of profile  $\lambda : i_1, \dots, i_k \xrightarrow{\bullet} *$  for a functor  $T : \mathbf{Set}^n \rightarrow \mathbf{Set}$ , where  $i_1, \dots, i_k \leq n$ , is a natural transformation

$$\lambda : (\mathcal{Q} \circ P_{i_1}^{op}) \times \dots \times (\mathcal{Q} \circ P_{i_k}^{op}) \rightarrow \mathcal{Q} \circ T^{op}$$

between functors  $(\mathbf{Set}^n)^{op} \rightarrow \mathbf{Set}$ , where  $\mathcal{Q}$  denotes the contravariant powerset functor  $\mathbf{Set}^{op} \rightarrow \mathbf{Set}$  (i.e.  $\mathcal{Q}X = \mathcal{P}X$ , and  $\mathcal{Q}(f)(A) = f^{-1}[A]$ ) and  $P_i : \mathbf{Set}^n \rightarrow \mathbf{Set}$  is the  $i$ -th projection.

We now construct a compositional coalgebraic semantics of the logic  $\mathcal{F}(\mathbb{G})$  induced by a gluing  $\mathbb{G}$  from structures associated with the features combined by  $\mathbb{G}$ . We first describe the notion of structure associated with a single feature, and then the combination of such structures along a gluing.

**Definition 7.** Let  $F = (\Lambda, \mathcal{R})$  be an  $n$ -ary feature. A *structure* for  $F$  consists of an endofunctor  $\llbracket F \rrbracket : \mathbf{Set}^n \rightarrow \mathbf{Set}$  and an assignment of a predicate lifting  $\llbracket L \rrbracket : i_1, \dots, i_k \xrightarrow{\bullet} *$  for  $T$  to every modal operator  $L : i_1, \dots, i_k \rightarrow *$  in  $\Lambda$ , subject to the condition that every rule  $R = \phi_1; \dots; \phi_n / \psi$  over  $V$  in  $\mathcal{R}$  is *one-step sound*: for every  $n$ -sorted set  $X$  and every assignment  $\tau$  of subsets  $\tau(a) \subseteq X_i$  to the variables  $a \in V_i$ , if  $\llbracket \phi_i \rrbracket \tau = X_i$  for all  $i$ , then  $\llbracket \psi \rrbracket \tau = TX$ , where  $\llbracket \phi_i \rrbracket \tau \subseteq X_i$  and  $\llbracket \psi \rrbracket \tau \subseteq TX$  are defined by the usual clauses for boolean operators and  $\llbracket L(a_1, \dots, a_k) \rrbracket \tau = \llbracket L \rrbracket (\tau(a_1), \dots, \tau(a_k))$ .

When features are equipped with structures, every feature expression  $t$  over the set  $\mathcal{S}$  of sorts defines a functor  $\llbracket t \rrbracket : \mathbf{Set}^{\mathcal{S}} \rightarrow \mathbf{Set}$  by

$$\llbracket a \rrbracket = P_a : \mathbf{Set}^{\mathcal{S}} \rightarrow \mathbf{Set} \quad (a \in \mathcal{S}) \quad \text{and} \quad \llbracket F(t_1, \dots, t_n) \rrbracket = \llbracket F \rrbracket \circ \langle \llbracket t_1 \rrbracket, \dots, \llbracket t_n \rrbracket \rangle,$$

where  $P_a$  is projection to the  $a$ -th component and  $\langle \cdot \rangle$  represents tupling. Thus, a gluing  $\mathbb{G} = (t_a)_{a \in \mathcal{S}}$  induces a functor  $\llbracket \mathbb{G} \rrbracket : \mathbf{Set}^{\mathcal{S}} \rightarrow \mathbf{Set}^{\mathcal{S}}$ .

The coalgebraic semantics of  $\mathcal{F}(\mathbb{G})$  is now given w.r.t.  $\llbracket \mathbb{G} \rrbracket$ -coalgebras  $C = (X, \xi)$ . For a type  $s \in \text{Types}(\mathbb{G})$ , an  $s$ -state of  $C$  is an element  $x \in \llbracket s \rrbracket X$ . The semantics of a formula  $\phi : s$  is a set  $\llbracket \phi \rrbracket_C \subseteq \llbracket s \rrbracket X$  of  $s$ -states. We have the usual clauses for propositional connectives, and the semantics of modal operators is given by the following

clauses for composite types (top, assuming  $F(s_1, \dots, s_n) \in \text{Types}(G)$ ) and base types (bottom, for  $a \rightarrow F(s_1, \dots, s_n) \in G$ ):

$$\begin{aligned} \llbracket L(\phi_1, \dots, \phi_n) : F(s_1, \dots, s_n) \rrbracket_C &= \llbracket L \rrbracket(\llbracket \phi_1 \rrbracket_C, \dots, \llbracket \phi_n \rrbracket_C) \\ \llbracket L(\phi_1, \dots, \phi_n) : a \rrbracket_C &= \xi_a^{-1} \circ \llbracket L \rrbracket(\llbracket \phi_1 \rrbracket_C, \dots, \llbracket \phi_n \rrbracket_C) \end{aligned}$$

where, in both cases,  $L : i_1, \dots, i_n \rightarrow *$  in  $F$ . We write  $x \models_C^s \phi$  if  $\phi : s$  and  $x \in \llbracket \phi \rrbracket_C$ .

Note that the requirement that rules are one-step sound immediately yields soundness of the logic w.r.t. the semantics described above; this is as in [4].

**Example 8.** The standard semantics for the features of Example 2 is induced by the following structures.

**Non-Determinism:** A structure for  $N_A$  is given by  $\llbracket N_A \rrbracket = \mathcal{P}(A \times \_)$  and

$$\llbracket \square_a \rrbracket_X(C) = \{B \in \mathcal{P}(A \times X) \mid \{x : (a, x) \in B\} \subseteq C\}.$$

Note that (single-sorted) coalgebras for  $\mathcal{P}(A \times \_)$  are labelled transition systems, and the lifting associated with  $\square_a$  gives rise to Hennessy-Milner logic [17].

**Uncertainty:** Put  $\llbracket U \rrbracket = D_\omega$ , where  $D_\omega$  is the *finite distribution functor*  $D_\omega$  that maps a set  $X$  to the set of probability distributions on  $X$  with finite support. The modal operators  $L_p$  are interpreted by

$$\llbracket L_p \rrbracket_X(A) = \{P \in D_\omega X \mid PA \geq p\}.$$

(Single-sorted)  $D_\omega$ -coalgebras are finitely branching probabilistic transition systems. For  $G = (s \rightarrow N_A(U(s)))$  (Example 4), we have  $\llbracket G \rrbracket = \mathcal{P} \circ D_\omega$ , so that  $\llbracket G \rrbracket$ -coalgebras are precisely Segala systems, while coalgebras for the corresponding flat signature have an explicit separation between non-deterministic and probabilistic states.

**Choice:** Let  $\llbracket S \rrbracket$  be the disjoint sum functor  $\llbracket S \rrbracket(X, Y) = X + Y$ , and interpret the modality  $\text{+}$  by

$$\llbracket + \rrbracket_{X,Y}(A, B) = A + B \subseteq X + Y.$$

**Fusion:** Let  $\llbracket P \rrbracket$  be the binary product functor  $\llbracket P \rrbracket(X, Y) = X \times Y$ , and put

$$\llbracket \pi_1 \rrbracket_{X,Y} A = \{(x, y) \mid x \in A\} \text{ and } \llbracket \pi_2 \rrbracket_{X,Y} B = \{(x, y) \mid y \in B\}.$$

**Conditionality:** Define the functor  $\llbracket C \rrbracket$  by  $\llbracket C \rrbracket(X, Y) = \mathcal{Q}X \rightarrow Y$ , with  $\mathcal{Q}$  denoting contravariant powerset and  $\rightarrow$  denoting function space, and put

$$\llbracket \overset{\bullet}{\Rightarrow} \rrbracket_{X,Y}(A, B) = \{f : \mathcal{Q}X \rightarrow Y \mid f(A) \in B\}.$$

For  $G = (c \rightarrow C(c, K(c)))$  (Example 4), we have  $\llbracket G \rrbracket X = \mathcal{Q}X \rightarrow \mathcal{P}X$ , and  $\llbracket G \rrbracket$ -coalgebras are conditional frames [3].

Modal logic talks only about the observable behaviour of states; this is formally expressed as invariance of the logic under morphisms:

**Proposition 9.** *Let  $f : C \rightarrow D$  be a morphism of  $\llbracket G \rrbracket$ -coalgebras. Then for each  $G$ -formula  $\phi : s$  and each  $s$ -state  $x$  in  $C$ ,  $x \models_C^s \phi$  iff  $(\llbracket s \rrbracket f)(x) \models_D^s \phi$ .*

We can now state the (local) *satisfiability problem* for multi-sorted modal logics.

**Definition 10.** A G-formula  $\phi : s$  is *satisfiable in a G-model* if there exist a  $\llbracket G \rrbracket$ -coalgebra  $C$  and an  $s$ -state  $x$  in  $C$  such that  $x \models_C^s \phi$ .

A central contribution of this work is to show that for every gluing, we can construct a flat gluing with an equivalent satisfiability problem. For flat gluings, one can generalise existing model constructions and complexity results for coalgebraic modal logic [22,23,21], and the relevant criteria reduce to the component logics; for the shallow-model-based *PSPACE* algorithm of [23], this is discussed in more detail in Sect. 5. We thus obtain compositional algorithmic methods also for the standard single-sorted semantics present in the literature.

We start by constructing a flattening  $G^b$  of an arbitrary gluing  $G$  and then transform  $\llbracket G \rrbracket$ -coalgebras to  $\llbracket G^b \rrbracket$ -coalgebras preserving satisfaction of formulas.

**Definition 11.** Let  $G$  be a gluing over the set  $\mathcal{S}$  of sorts. The *flattening*  $G^b = (u_s)_{s \in \text{Types}(G)}$  of  $G$  is a flat gluing over the set  $\mathcal{S}^b = \text{Types}(G)$  of sorts, defined by  $u_s = t_a$  for  $s = a \in \mathcal{S}$  (with immediate subexpressions of  $t_a$  regarded as sorts in  $\mathcal{S}^b$ ) and  $u_s = s$  otherwise (with  $s$  regarded as a sort in  $\mathcal{S}^b$ ).

**Example 12.** Given the gluings  $G_1 \equiv (a \rightarrow S(U(a), N_A(a)))$  and  $G_2 \equiv (a \rightarrow S(u, n), u \rightarrow U(a), n \rightarrow N_A(a))$  from Example 4,  $G_2$  is the flattening of  $G_1$ , up to renaming the *sorts*  $U(a)$  and  $N_A(a)$  of the flattening into  $u$  and  $n$ , respectively.

It is easy to see that *the flattening  $G^b$  syntactically induces the same logic as  $G$* , i.e. the types, formulas, and proof systems coincide. Our main result is now stated as follows.

**Theorem 13.** A G-formula is *satisfiable in a G-model* iff it is *satisfiable (as a  $G^b$ -formula) in a  $G^b$ -model*.

*Proof.* (Sketch) ‘*Only if*’: Expand a  $\llbracket G \rrbracket$ -coalgebra  $C$  to a  $\llbracket G^b \rrbracket$ -coalgebra  $C^b$  by inserting identity functions for the components of the structure map corresponding to composite types. Induction on the definition of the semantics then shows that the semantics w.r.t.  $C$  and  $C^b$  agree.

‘*If*’: Turn a  $\llbracket G^b \rrbracket$ -coalgebra  $D = (X_b, \xi_b)_{b \in \mathcal{S}^b}$  into the  $\llbracket G \rrbracket$ -coalgebra  $D^\sharp = (X_a, \gamma_a)_{a \in \mathcal{S}}$ , where

$$\gamma_a = \llbracket F \rrbracket(\zeta_{s_1}, \dots, \zeta_{s_n}) \circ \xi_a$$

for  $a \rightarrow F(s_1, \dots, s_n)$  in  $G$ , and the maps  $\zeta_s : X_s \rightarrow \llbracket s \rrbracket(X_a)_{a \in \mathcal{S}}$  for  $s \in \text{Types}(G)$  are defined recursively by

$$\zeta_a = id_{X_a} \quad (a \in \mathcal{S}) \quad \text{and} \quad \zeta_{F(s_1, \dots, s_n)} = \llbracket F \rrbracket(\zeta_{s_1}, \dots, \zeta_{s_n}) \circ \xi_{F(s_1, \dots, s_n)}.$$

One can then construct a coalgebra morphism  $D \rightarrow (D^\sharp)^b$ , and Proposition 9 yields the claim.  $\square$

In our running example, the situation is as follows:

**Example 14.** Consider the gluings  $G_1$  and  $G_2$  over  $\mathcal{S} = \{a, u, n\}$  from Example 4 and recall from Example 12 that  $G_1^b = G_2$ . Let  $C = (X, \xi : X \rightarrow D_\omega X + \mathcal{P}(A \times X))$  be a  $\llbracket G_1 \rrbracket$ -coalgebra. Then  $C^b = ((X_s), (\xi_s))$  where  $X_a = X, X_u = D_\omega X, X_n = \mathcal{P}(A \times X), \xi_a = s, \xi_u = id_{X_u}$ , and  $\xi_n = id_{X_n}$ .

Conversely, given a  $\llbracket G_2 \rrbracket$ -coalgebra  $D = ((X_s), (\xi_s))$ , we construct a  $\llbracket G_1 \rrbracket$ -coalgebra  $D^\sharp = (X, \xi)$  by putting  $X = X_a$  and  $\xi = (\xi_u + \xi_n) \circ \xi_a$ . The triple  $(id_X, \xi_u, \xi_s)$  is a homomorphism  $D \rightarrow (D^\sharp)^b$ .

## 5 Applications to Model Construction and Complexity

We have seen in Sect. 3 that the same multi-sorted logic can arise from different gluings of given features, where the difference manifests itself only on a semantic level. The different interpretations of the logic are related by Theorem 13 which shows that the satisfiability problem for a given gluing is equivalent to that of its *flattening*. We now show that the generic shallow model construction and the ensuing *PSPACE* decision procedure from [23] generalise to flat gluings; this enables us to derive upper *PSPACE* bounds for arbitrary gluings, in particular for heterogeneous logics equipped with their standard single-sorted semantics as in Sect. 2.

The shallow model construction requires the involved structures to be strictly one-step complete in the following sense, where the notation  $\llbracket - \rrbracket_\tau$ , is as in Definition 7 (Strict) one-step completeness implies weak completeness of the rule system [17|22].

**Definition 15.** An  $n$ -ary feature  $F$  is *strictly one-step complete* for a structure  $T = \llbracket F \rrbracket : \mathbf{Set}^n \rightarrow \mathbf{Set}$  if, whenever  $\llbracket \chi \rrbracket_\tau = T(X_1, \dots, X_n)$  for a sorted set  $(V_1, \dots, V_n)$  of variables, an assignment  $\tau$  of subsets  $\tau(a) \subseteq X_i$  to variables  $a \in V_i$ , and a clause  $\chi$  over atoms of the form  $L(a_{i_1}, \dots, a_{i_k})$ , where  $L : i_1, \dots, i_k \rightarrow *$  in  $F$  and  $a_{i_j} \in V_{i_j}$ , then  $\chi$  is propositionally entailed by a clause  $\psi\sigma$ , where  $(\phi_i)/\psi$  is a rule of  $F$  and  $\sigma$  is a  $(V_1, \dots, V_n)$ -substitution (i.e.  $\sigma(a) \in V_i$  for  $a \in V_i$ ) such that  $\llbracket \phi_i\sigma \rrbracket_\tau = X_i$  for all  $i$ .

(The formulation above corrects the formulation given in [23] in admitting only a single rule application in a strict derivation.) One shows analogously to the single-sorted case [22] that the set of all one-step sound rules for a given  $F$ -structure is strictly one-step complete, so that strictly one-step complete axiomatisations always exist. In [23], *rule resolution*, a systematic procedure for obtaining strictly one-step complete rule sets, has been described, which straightforwardly generalises to the multi-sorted setting. Throughout this section, we fix a gluing  $G$  of a set  $\Phi$  of features over a set  $\mathcal{S}$  of sorts; moreover we assume that every feature is equipped with a structure.

**Definition 16.** The set  $MA(\phi)$  of *modal atoms* of an  $G$ -formula  $\phi$  is defined recursively by  $MA(\phi \wedge \psi) = MA(\phi) \cup MA(\psi)$ ,  $MA(\neg\phi) = MA(\phi)$ , and  $MA(L(\rho_1, \dots, \rho_n)) = \{L(\rho_1, \dots, \rho_n)\}$ . A *pseudovaluation* for  $\phi$  is a subset  $H$  of  $MA(\phi)$ . We define satisfaction of propositional formulas  $\chi$  over  $MA(\phi)$  by  $H (H \models \chi)$  inductively in the obvious way, with  $H \models \chi \iff \chi \in H$  for  $\chi \in MA(\phi)$ .

Assuming that  $s = F(s_1, \dots, s_n) \in \mathbf{Types}(G)$  if  $s$  is composite and  $a \rightarrow F(s_1, \dots, s_n) \in G$  if  $s = a$  is a base type, we say that a rule  $R = (\phi_1; \dots; \phi_n)/\psi$

associated with the feature  $F$  matches a pseudovaluation  $H$  for  $\phi : s$  if there is a substitution  $\sigma$  such that  $\psi\sigma$  is a clause over  $MA(\phi)$  with  $H \not\models \psi\sigma$ . In this case, the pair  $(R, \sigma)$  is called a *matching* of  $H$ .

Our shallow model theorem now takes the following form.

**Theorem 17.** *If every feature in  $G$  is strictly one-step complete, then a formula  $\phi : s$  is satisfiable in a  $G$ -model iff  $H \models \phi$  for some pseudovaluation  $H$  for  $\phi$  such that for every matching  $((\phi_1; \dots; \phi_n)/\psi, \sigma)$  of  $H$ , one of the formulas  $\neg\phi_i\sigma$  is satisfiable.*

The proof first reduces to flat gluings by Theorem 13 and then recursively constructs a shallow model whose root state is a pseudovaluation and whose branches are models of negated substitution instances of rule premises as in the statement.

From Theorem 17 we obtain a multi-sorted version of the generic *PSPACE* decision procedure of 23. This requires to compute matchings of given pseudovaluations, and we require that the rules associated with features are *reduction closed*, i.e. it suffices to consider matchings  $((\phi_1, \dots, \phi_n)/\psi, \sigma)$  where  $\psi\sigma$  does not contain duplicate literals, with the consequence that there are only finitely many matches to check in every recursion step. Since rules are generally too large to pass around directly, we assume that every rule is represented by a *code*, i.e. a string over some alphabet. For the features discussed in Example 2 the codes can be taken as the parameters of the rules.

The crucial requirement for the effectivity of the algorithm is that one has a polynomial bound on codes of matching rules and that a number of minor infrastructure operations can be performed in polynomial time (cf. 23 for details), in which case we call a rule set *PSPACE-tractable*. All rule sets presented in Fig. 1 are strictly one-step complete, reduction closed, and *PSPACE-tractable* (this is either clear or shown in 23). We obtain:

**Theorem 18 (Space Complexity).** *If every feature in  $G$  is strictly one-step complete, reduction closed, and *PSPACE-tractable*, then the satisfiability problem for  $\mathcal{F}(G)$ -formulas over  $\llbracket G \rrbracket$ -coalgebras is in *PSPACE*.*

In particular, satisfiability for logics arising through arbitrary gluings of the features from Example 2 are in *PSPACE*.

**Remark 19.** The recursive structure of the algorithm allows for a modular implementation which interconnects separate matching routines for each feature. In particular, the same algorithmic structure may alternatively be applied to effective heuristic matching routines, leading to approximative but more efficient solutions.

## 6 Conclusions

We have introduced a calculus of *gluings*, which describe ways of combining logical features like uncertainty, non-determinism, and choice. We have shown that the satisfaction problem of a gluing is in *PSPACE* if this is true for involved features. This has been achieved by equipping the logics under consideration with a multi-sorted coalgebraic semantics. Crucially, we have shown that the satisfiability problem of a gluing is equivalent to that of a corresponding flattened gluing; flat gluings are technically more

tractable than general gluings, in particular allow for a straightforward generalization of the generic algorithm of [23]. Our results pave the way for modularized tool support for a large class of heterogeneous logics. The study of  $\mathcal{E}$ -connections [14] in the coalgebraic framework is the subject of future work.

## References

1. Bartels, F., Sokolova, A., de Vink, E.: A hierarchy of probabilistic system types. In: Gumm, H.-P. (ed.) *Coalgebraic Methods in Computer Science*. ENTCS, vol. 82, Elsevier, Amsterdam (2003)
2. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Cambridge University Press, Cambridge (2001)
3. Chellas, B.: *Modal Logic*. Cambridge University Press, Cambridge (1980)
4. Cîrstea, C., Pattinson, D.: Modular construction of modal logics. *Theoret. Comput. Sci.* (to appear). Earlier version In: Gardner, P., Yoshida, N. (eds.) *CONCUR 2004*. LNCS, vol. 3170, pp. 258–275. Springer, Heidelberg (2004)
5. D’Agostino, G., Visser, A.: Finality regained: A coalgebraic study of Scott-sets and multisets. *Arch. Math. Logic* 41, 267–298 (2002)
6. Fine, K.: In so many possible worlds. *Notre Dame J. Formal Logic* 13, 516–520 (1972)
7. Halpern, J.Y.: *Reasoning About Uncertainty*. MIT Press, Cambridge (2003)
8. Hansson, H., Jonsson, B.: A calculus for communicating systems with time and probabilities. In: *Real-Time Systems, RTSS 90*, pp. 278–287. IEEE Computer Society Press, Los Alamitos (1990)
9. Heifetz, A., Mongin, P.: Probabilistic logic for type spaces. *Games and Economic Behavior* 35, 31–53 (2001)
10. Hemaspaandra, E.: Complexity transfer for modal logic. In: Abramsky, S. (ed.) *Logic in Computer Science, LICS ’94*, pp. 164–173. IEEE Computer Society Press, Los Alamitos (1994)
11. Jacobs, B.: Many-sorted coalgebraic modal logic: a model-theoretic study. *Theor. Inform. Appl.* 35, 31–59 (2001)
12. Jonsson, B., Yi, W., Larsen, K.G.: Probabilistic extensions of process algebras. In: Bergstra, J., Ponse, A., Smolka, S.M. (eds.) *Handbook of Process Algebra*, Elsevier, Amsterdam (2001)
13. Kurucz, A.: Combining modal logics. In: van Benthem, J., Blackburn, P., Wolter, F. (eds.) *Handbook of Modal Logic*, Elsevier, Amsterdam (2006)
14. Kutz, O., Lutz, C., Wolter, F., Zakharyashev, M.:  $\mathcal{E}$ -connections of abstract description systems. *Artificial Intelligence* 156, 1–73 (2004)
15. Larsen, K., Skou, A.: Bisimulation through probabilistic testing. *Inform. Comput.* 94, 1–28 (1991)
16. Mossakowski, T., Schröder, L., Roggenbach, M., Reichel, H.: Algebraic-coalgebraic specification in CoCASL. *J. Logic Algebraic Programming* 67, 146–197 (2006)
17. Pattinson, D.: Expressive logics for coalgebras via terminal sequence induction. *Notre Dame J. Formal Logic* 45, 19–33 (2004)
18. Pauly, M.: A modal logic for coalitional power in games. *J. Logic Comput.* 12, 149–166 (2002)
19. Rutten, J.: Universal coalgebra: A theory of systems. *Theoret. Comput. Sci.* 249, 3–80 (2000)
20. Schröder, L.: Expressivity of coalgebraic modal logic: the limits and beyond. *Theoret. Comput. Sci.* (to appear). Earlier version In: Sassone, V. (ed.) *FOSSACS 2005*. LNCS, vol. 3441, pp. 440–454. Springer, Heidelberg (2005)

21. Schröder, L.: A semantic PSPACE criterion for the next 700 rank 0-1 modal logics. Available at <http://www.informatik.uni-bremen.de/~lschrode/papers/rank01pspace.pdf>
22. Schröder, L.: A finite model construction for coalgebraic modal logic. In: Aceto, L., Ingólfssdóttir, A. (eds.) FOSSACS 2006 and ETAPS 2006. LNCS, vol. 3921, pp. 157–171. Springer, Heidelberg (2006)
23. Schröder, L., Pattinson, D.: PSPACE reasoning for rank-1 modal logics. In: Alur, R. (ed.) Logic in Computer Science, LICS '06, pp. 231–240. IEEE Computer Society Press, Los Alamitos (2006)
24. Segala, R.: Modelling and Verification of Randomized Distributed Real-Time Systems. PhD thesis, Massachusetts Institute of Technology (1995)
25. Tobies, S.: PSPACE reasoning for graded modal logics. *J. Logic Comput.* 11, 85–106 (2001)
26. Wolter, F.: Fusions of modal logics revisited. In: Zakharyashev, M., Segerberg, K., de Rijke, M., Wansing, H. (eds.) *Advances in modal logic*. CSLI Lect. Notes, vol. 1, pp. 361–379. CSLI, Stanford (1998)



# Co-Logic Programming: Extending Logic Programming with Coinduction

Luke Simon, Ajay Bansal, Ajay Mallya, and Gopal Gupta

Department of Computer Science  
University of Texas at Dallas, Richardson, TX 75080

**Abstract.** In this paper we present the theory and practice of *co-logic programming* (co-LP for brevity), a paradigm that combines both inductive and coinductive logic programming. Co-LP is a natural generalization of logic programming and coinductive logic programming, which in turn generalizes other extensions of logic programming, such as infinite trees, lazy predicates, and concurrent communicating predicates. Co-LP has applications to rational trees, verifying infinitary properties, lazy evaluation, concurrent LP, model checking, bisimilarity proofs, etc.

## 1 Introduction

Recently *coinductive logic programming* has been introduced as a means of programming with infinite data structures and supporting *co-recursion* [2] in logic programming [9]. Practical applications of coinductive LP include improved modularization of programs as seen in lazy functional languages, rational terms, and model checking. Coinductive LP allows infinite proofs and infinite data structures via a declarative semantics based on greatest fix points and an operational semantics based on the *coinductive hypothesis rule* that recognizes a *co-recursive* call and succeeds.

There are problems for which coinductive LP is better suited than traditional inductive LP (not to be confused with ILP, LP systems that deal with machine learning). Conversely, there are problems for which inductive LP is better suited than coinductive LP. But there are even more problems where both coinductive and inductive logic programming paradigms are simultaneously useful. In this paper we examine the combination of coinductive and inductive LP. We christen the new paradigm *co-logic programming* (co-LP for brevity). *Thus, co-LP subsumes both coinductive LP and inductive LP.* A combination of inductive and coinductive LP is not straightforward as cyclical nesting of inductive and coinductive definitions results in programs to which proper semantics cannot be given. *Co-logic programming* combines traditional and coinductive LP by allowing predicates to be optionally annotated as being coinductive; by default, unannotated predicates are interpreted as inductive. In our formulation of co-LP, coinductive predicates can call inductive predicates and *vice versa*, with the only exception being that no cycles are allowed through alternating calls to inductive and coinductive predicates. This results in a natural generalization of

logic programming and coinductive logic programming, which in turn generalizes other extensions of logic programming, such as infinite trees, lazy predicates, and concurrent communicating predicates. In this paper the declarative semantics for co-LP is defined, and a corresponding top-down, goal-directed operational semantics is provided in terms of alternating SLD and co-SLD semantics and proved equivalent to the declarative semantics.

Applications of co-LP are also discussed. Co-LP has applications to rational trees, verifying infinitary properties, lazy evaluation, concurrent logic programming, model checking, bisimilarity proofs, Answer Set Programming (ASP), etc. Our work can be thought of as developing a practical and reasonable top-down operational semantics for computing the alternating least and greatest fixed-point of a logic program. We assume that the reader has familiarity with coinduction [2] & coinductive LP [9].

## 2 Coinductive Logic Programming

Coinductive LP allows one to program with infinite data structures and infinite proofs. Under Coinductive LP, programs such as the one below, which describes infinite streams of binary digits, become semantically meaningful, i.e., not semantically null.

```
bit(0).
bit(1).
bitstream([H|T]) :- bit(H), bitstream(T).

| ?- X = [0, 1, 1, 0 | X], bitstream(X).
```

We would like the above query to have a finite derivation and return a positive answer; however, aside from the `bit` predicate, the least fixed-point (lfp) semantics of the above program is null, and its evaluation using SLD resolution lacks a finite derivation. The problems are two-fold. The Herbrand universe does not allow for infinite terms such as `X` and the least Herbrand model does not allow for infinite proofs, such as the proof of `bitstream(X)`; yet these concepts are commonplace in computer science, and a sound mathematical foundation exists for them in the field of hyperset theory [2]. Coinductive LP extends the traditional declarative and operational semantics of LP to allow reasoning over infinite and cyclic structures and properties [9].

In the coinductive LP paradigm the declarative semantics of the predicate `bitstream/1` above is given in terms of *infinitary Herbrand universe*, *infinitary Herbrand base*, and *maximal models (computed using greatest fixed-points)*. The operational semantics is given in terms of the *coinductive hypothesis rule* which states that during execution, if the current resolvent  $R$  contains a call  $C'$  that unifies with a call  $C$  encountered earlier, then the call  $C'$  succeeds; the new resolvent is  $R'\theta$  where  $\theta = mgu(C, C')$  and  $R'$  is obtained by deleting  $C'$  from  $R$ . With this extension a clause such as `p([1|T]) :- p(T)` and the query `p(Y)` will produce an infinite answer  $Y = [1|Y]$ . Applications of purely coinductive

logic programming to fields such as model checking, concurrent logic programming, real-time systems, etc., can also be found in [9]. To implement coinductive LP, one needs to remember in a memo-table (memoize) all the calls made to coinductive predicates.

### 3 Motivation and Examples

Coinductive LP and inductive LP cannot be naively combined together, as this results in interleaving of least fixed point and greatest fixed point computations. Such programs cannot be given meaning easily. Consider the following program:

```
:- coinductive p/0.    %q/0 is inductive by default.
p :- q.
q :- p.
```

For computing the result of goal `?- q.`, we'll use `lfp` semantics, which will produce null, implying that `q` should fail. Given the goal `?- p.` now, it should also fail, since `p` calls `q`. However, if we use `gfp` semantics (and the coinductive hypothesis computation rule), the goal `p` should succeed, which, in turn, implies that `q` should succeed. Thus, naively mixing coinduction and induction leads to contradictions.

We resolve this contradiction by disallowing such cyclical nesting of inductive and coinductive predicates. Thus, inductive and coinductive predicates can be used in the same program as long as the program is *stratified* w.r.t. inductive and coinductive predicates. That is, an inductive predicate in a given strata cannot call a coinductive predicate in a higher strata and vice versa.

Next, we illustrate co-LP via more examples.

**Infinite Streams:** The following example involves a combination of an inductive predicate and a coinductive predicate. By default, predicates are inductive, unless indicated otherwise. Consider the execution of the following program, which defines a predicate that recognizes infinite streams of natural numbers. Note that only the `stream/1` predicate is coinductive, while the `number/1` predicate is inductive.

```
:- coinductive stream/1.
stream( [ H | T ] ) :- number( H ), stream( T ).
number( 0 ).
number( s(N) ) :- number( N ).
| ?- stream( [ 0, s(0), s(s(0)) | T ] ).
```

The following is an execution trace, for the above query, of the memoization of calls by the operational semantics. Note that calls of `number/1` are not memo'ed because `number/1` is inductive.

```
MEMO: stream( [ 0, s(0), s(s(0)) | T ] )
MEMO: stream( [ s(0), s(s(0)) | T ] )
MEMO: stream( [ s(s(0)) | T ] )
```

The next goal call is `stream(T)`, which unifies with the first memo'ed ancestor, and therefore immediately succeeds. Hence the original query succeeds with the infinite solution:

$$T = [ 0, s(0), s(s(0)) \mid T ]$$

The user could force a failure here, which would cause the goal to be unified with the next two matching memo'ed ancestor producing  $T = [s(0), s(s(0)) \mid T]$  &  $T = [s(s(0)) \mid T]$  respectively. If no remaining memo'ed elements exist, the goal is memo'ed, and expanded using the coinductively defined clauses, and the process repeats—generating additional results, and effectively enumerating the set of (rational) infinite lists of natural numbers that begin with the prefix  $[0, s(0), s(s(0))]$ .

The goal `stream(T)` is true whenever  $T$  is some infinite list of natural numbers. If `number/1` was also coinductive, then `stream(T)` would be true whenever  $T$  is a list containing either natural numbers or  $\omega$ , i.e., infinity, which is represented as an infinite application of successor  $s(s(s(\dots)))$ . Such a term has a finite representation as  $X = s(X)$ .

Note that excluding the occurs check is necessary as such structures have a greatest fixed-point interpretation and are in the co-Herbrand Universe. This is in fact one of the benefits of co-LP. Unification without occurs check is typically more efficient than unification with occurs check, and now it is even possible to define non-trivial predicates on the infinite terms that result from such unification, which are not definable in LP with rational trees. Traditional logic programming's least Herbrand model semantics requires SLD resolution to unify with occurs check (or lack soundness), which adversely affects performance in the common case. Co-LP, on the other hand, has a declarative semantics that allows unification without doing occurs check, and it also allows for non-trivial predicates to be defined on infinite terms resulting from such unification.

**List Membership:** This example illustrates that some predicates are naturally defined inductively, while other predicates are naturally defined coinductively. The `member/2` predicate is an example of an inherently inductive predicate.

```
member( H, [ H | _ ] ).
member( H, [ _ | T ] ) :- member( H, T ).
```

If this predicate was declared to be coinductive, then `member( X, L )` is true (i) whenever  $X$  is in  $L$  or (ii) whenever  $L$  is an infinite list, even if  $X$  is not in  $L$ ! The definition above, whether declared coinductive or not, states that the desired element is the last element of some prefix of the list, as the following equivalent reformulation of `member/2`, called `membera/2` shows, where `drop/3` drops a prefix ending in the desired element and returns the resulting suffix.

```
membera( X, L ) :- drop( X, L, _ ).
drop( H, [ H | T ], T ).
drop( H, [ _ | T ], T1 ) :- drop( H, T, T1 ).
```

When the predicate is inductive, this prefix must be finite, but when the predicate is declared coinductive, the prefix may be infinite. Since an infinite list has no last element, it is trivially true that the last element unifies with any other

term. This explains why the above definition, when declared to be coinductive, is always true for infinite lists regardless of the presence of the desired element.

A mixture of inductive and coinductive predicates can be used to define a variation of `member/2`, called `comember/2`, which is true if and only if the desired element occurs an infinite number of times in the list. Hence it is false when the element does not occur in the list or when the element only occurs a finite number of times in the list. On the other hand, if `comember/2` was declared inductive, then it would always be false. Hence coinduction is a necessary extension.

```
:- coinductive comember/2.
comember( X, L ) :- drop( X, L, L1 ), comember( X, L1 ).
?- X = [ 1, 2, 3 | X ], comember( 2, X ).
    Answer: yes.
?- X = [ 1, 2, 3, 1, 2, 3 ], comember( 2, X ).
    Answer: no.
?- X = [ 1, 2, 3 | X ], comember( Y, X ).
    Answer: Y = 1;
           Y = 2;
           Y = 3;
```

Note that `drop/3` will have to be evaluated using OLDT tabling for it not to go into an infinite loop for inputs such as `X = [1,2,3|X]` (if `X` is absent from the list `L`, the `lfp` of `drop(X,L)` is null). More elaborate examples including application to model checking can be found elsewhere [\[10\]](#).

## 4 Syntax and Semantics

While co-logic programming is based on the very simple concept of co-induction, the previous examples show that it is an extremely elegant and powerful paradigm. Next, we present the declarative and operational semantics of co-logic programming and prove that they are equivalent.

**Syntax:** A co-logic program  $P$  is syntactically identical to a traditional, that is, inductive logic program. In the following, it is important to distinguish between an idealized class of objects and the syntactic restriction of said objects. Elements of syntax are necessarily finite, while many of the semantic objects used by co-LP are infinite. It is assumed that there is an enumerable set of variables, an enumerable set of constants, and for all natural numbers  $n$ , there are an enumerable set of function and predicate symbols of arity  $n$ .

**Definition 1.** *A pre-program is a definite program paired with a mapping of predicate symbols to the token `coinductive` or `inductive`. A predicate is `coinductive` (resp. `inductive`) if the partial mapping maps the predicate to `coinductive` (resp. `inductive`). An atom is `coinductive` (resp. `inductive`) if the underlying predicate is `coinductive` (resp. `inductive`).*

Not every pre-program is a co-logic program. Co-LP programs do not allow for any pair of inductive and coinductive predicates to be mutually recursive:

programs must be stratified w.r.t. alternating induction and coinduction. An inductive predicate can (indirectly) call a coinductive predicate and visa versa, but they cannot be mutually recursive.

**Definition 2.** *In some pre-program  $P$ , we say that a predicate  $p$  depends on a predicate  $q$  if and only if  $p = q$  or  $P$  contains a clause  $C \leftarrow D_1, \dots, D_n$  such that  $C$  contains  $p$  and some  $D_i$  contains  $q$ . The dependency graph of program  $P$  has the set of its predicates as vertices, and the graph has an edge from  $p$  to  $q$  if and only if  $p$  depends on  $q$ .*

Co-LP programs are pre-programs with stratification restriction.

**Definition 3.** *A co-logic program is a pre-program such that for any strongly connected component  $G$  in the dependency graph of the program, every predicate in  $G$  is either mapped to *coinductive* or to *inductive*.*

**Declarative Semantics:** The declarative semantics of a co-logic program is a stratified interleaving of the minimal Herbrand model [17] and the maximal co-Herbrand model semantics [9]. Hence, co-LP strictly contains logic programming with rational trees [6] as well as coinductive logic programming [9]. This allows the universe of terms to contain infinite terms, in addition to the traditional finite terms. Finally, co-LP also allows for the model to contain ground goals that have either finite or infinite proofs. Note that stratification is necessary because programs that cyclically interleave inductive and coinductive predicates cannot be given a meaning easily, as explained earlier.

The following definition is necessary for defining the model of a co-logic program. Intuitively, a reduced graph is derived from a dependency graph by collapsing the strongly connected components of the dependency graph into single nodes. The graph resulting from this process is acyclic.

**Definition 4.** *The reduced graph for a co-logic program has vertices consisting of the strongly connected components of the dependency graph of  $P$ . There is an edge from  $v_1$  to  $v_2$  in the reduced graph if and only if some predicate in  $v_1$  depends on some predicate in  $v_2$ . A vertex in a reduced graph is said to be *coinductive* (resp. *inductive*) if it contains only *coinductive* (resp. *inductive*) predicates.*

A vertex in a reduced graph of a program  $P$  is called a stratum, as the set of predicates in  $P$  is stratified into a collection of mutually disjoint strata of predicates. The stratification restriction states that all vertices in the same stratum are of the same kind, i.e., every stratum is either inductive or coinductive. A stratum  $v$  depends on a stratum  $v'$ , when there is an edge from  $v$  to  $v'$  in the reduced graph. When there is a path in the reduced graph from  $v$  to  $v'$ ,  $v$  is said to be higher than  $v'$  and  $v'$  is said to be lower than  $v$ , and the case when  $v \neq v'$  is delineated by the modifier “strictly”, as in “strictly higher” and “strictly lower”. This restriction allows for the model of a stratum  $v$  to be defined in terms of the models of the strictly lower strata, upon which  $v$  depends. However, before we can define the model of a stratum, we must define a few basic sets. In the description below,  $\mu$  (resp.  $\nu$ ) is the least (resp. greatest) fixed point operator.

**Definition 5.** Let  $P$  be a definite program. Let  $A(P)$  be the set of constants in  $P$ , and let  $F_n(P)$  denote the set of function symbols of arity  $n$  in  $P$ . The co-Herbrand universe of  $P$ , denoted  $U^{co}(P) = \nu\Phi_P$ , where

$$\Phi_P(S) = A(P) \cup \{f(t_1, \dots, t_n) \mid f \in F_n(P) \wedge t_1, \dots, t_n \in S\}$$

Intuitively, this is the set of terms both finite and infinite that can be constructed from the constants and functions in the program. Hence unification *without* occurs check has a greatest fixed-point interpretation, as rational trees are included in the co-Herbrand universe. The Herbrand universe is simply  $\mu\Phi_P$ .

**Definition 6.** Let  $P$  be a definite program. The co-Herbrand base a.k.a. the infinitary Herbrand base, written  $B^{co}(P)$ , is the set of all ground atoms that can be formed from the atoms in  $P$  and the elements of  $U^{co}(P)$ . Also, let  $G^{co}(P)$  be the set of ground clauses  $C \leftarrow D_1, \dots, D_n$  that are a ground instance of some clause of  $P$  such that  $C, D_1, \dots, D_n \in B^{co}(P)$ .

Now we can define the model of a stratum, i.e., the model of a vertex in the reduced graph of a co-logic program. The model of each stratum is defined using what is effectively the same  $T_P$  monotonic operator used in defining the minimal Herbrand model [17], except that it is extended so that it can treat the atoms defined as true in lower strata as facts when proving atoms containing predicates in the current stratum. This is possible because co-logic programs are stratified such that the reduced graph of a program is always a DAG and every predicate in the same stratum is the same kind: inductive or coinductive.

**Definition 7.** The model of a stratum  $v$  of  $P$  equals  $\mu T_P^v$  if  $v$  is inductive and  $\nu T_P^v$  if  $v$  is coinductive, such that  $R$  is the union of the models of the strata that are strictly lower than  $v$  and

$$T_P^v(S) = R \cup \left\{ q(\hat{t}) \mid q \in v \wedge [q(\hat{t}) \leftarrow \hat{D}] \in G^{co}(P) \wedge \hat{D} \in S \right\}$$

Since any predicate resides in exactly one stratum, the definition of the model of a co-logic program is straightforward.

**Definition 8.** The model of a co-logic program  $P$ , written  $M(P)$ , is the union of the model of every stratum of  $P$ .

Obviously co-LP's semantics subsumes the minimal co-Herbrand model used as the semantics for logic programming with rational trees, as well as the maximal co-Herbrand model used as the semantics for coinductive logic programming.

**Definition 9.** An atom  $A$  is true in program  $P$  iff the set of all groundings of  $A$  with substitutions ranging over the  $U^{co}(P)$ , is a subset of  $M(P)$ .

*Example 1.* Let  $P_1$  be the following program.

```
:- coinductive from/2.
from(N, [N|T]) :- from(s(N), T).
| ?- from(0, -).
```

The model of the program, which is defined in terms of an alternating fixed-point is as follows. The co-Herbrand Universe is  $U^{co}(P_1) = N \cup \Omega \cup L$  where  $N = \{0, s(0), s(s(0)), \dots\}$ ,  $\Omega = \{s(s(s(\dots)))\}$ , and  $L$  is the set of all finite and infinite lists of elements in  $N$ ,  $\Omega$ , and even  $L$ . Therefore the model  $M(P_1) = \{from(t, [t, s(t), s(s(t)), \dots]) \mid t \in U^{co}(P_1)\}$ , which is the meaning of the program and is obviously not null, as was the case with traditional logic programming. Furthermore  $from(0, [0, s(0), s(s(0)), \dots]) \in M(P_1)$  implies that the query returns “yes”. On the other hand, if the directive on the first line of the program was removed, call the resulting program  $P'_1$ , then the program’s only predicate would by default be inductive, and  $M(P'_1) = \emptyset$ . This corresponds to the traditional semantics of logic programming with infinite trees. Examples involving multiple strata of different kinds, i.e., mixing inductive and coinductive predicates, are given in section 3.

The model characterizes semantics in terms of truth, that is, the set of ground atoms that are true. This set is defined via a generator; later we will need to talk about the manner in which the generator is applied in order to include an atom in the model. For example, the generator is only allowed to be applied a finite number of times for any given atom in a least fixed-point, while it can be applied an infinite number of times in the greatest fixed-point. We capture this by recording the application of the generator in the elements of the fixed-point itself. We call these objects “idealized proofs.” In order to define idealized proofs, it is first necessary to define some formalisms for trees.

**Definition 10.** *A path  $\pi$  is a finite sequence of positive integers  $i$ . The empty path is written  $\epsilon$ , the singleton path is written  $i$  for some positive integer  $i$ , and the concatenation of two paths is written  $\pi \cdot \pi'$ . A tree of  $S$ , called an  $S$ -tree, is formally defined as a partial function from paths to elements of  $S$ , such that the domain is non-empty and prefix-closed. A node in a tree is unambiguously denoted by a path. So a tree  $t$  is described by the paths  $\pi$  from the root  $t(\epsilon)$  to the nodes  $t(\pi)$  of the tree, and the nodes are labeled with elements of  $S$ .*

*A child of node  $\pi$  in tree  $t$  is any path  $\pi \cdot i$  that is in the domain of  $t$ , where  $i$  is some positive integer. If  $\pi$  is in the domain of  $t$ , then the subtree of  $t$  rooted at  $\pi$ , written  $t \setminus \pi$ , is the partial function  $t'(\pi') = t(\pi \cdot \pi')$ . Also,  $node(L, T_1, \dots, T_n)$  denotes a constructor of an  $S$ -tree with root labeled  $L$  and subtrees  $T_i$ , where  $L \in S$  and each  $T_i$  is an  $S$ -tree, such that  $1 \leq i \leq n$ ,  $node(L, T_1, \dots, T_n)(\epsilon) = L$ , and  $node(L, T_1, \dots, T_n)(i \cdot \pi) = T_i(\pi)$ .*

Idealized proofs are trees of ground atoms, such that a parent is deduced from the idealized proofs of its children.

**Definition 11.** *The set of idealized proofs of a stratum of  $P$  equals  $\mu \Sigma_P^v$  if  $v$  is inductive and  $\nu \Sigma_P^v$  if  $v$  is coinductive, such that  $R$  is the union of the sets of idealized proofs of the strata strictly lower than  $v$  and*

$$\Sigma_P^v(S) = R \cup \{node(q(\hat{t}), T_1, \dots, T_n) \mid q \in v \wedge T_i \in S \wedge [q(\hat{t}) \leftarrow D_1, \dots, D_n] \in G^{co}(P) \wedge T_i(\epsilon) = D_i\}$$



Note that these definitions mirror the definitions defining models, with the exception that the elements of the sets record the application of the program clauses as a tree of atoms.

**Definition 12.** *The set of idealized proofs generated by a co-logic program  $P$ , written  $\Sigma_P$ , is the union of the sets of idealized proofs of every stratum of  $P$ .*

Again, this is nothing more than a reformulation of  $M(P)$ , which records the applications of the generator in the elements of the fixed-points, as the following theorem demonstrates.

**Theorem 1.** *Let  $S = \{A \mid \exists T \in \Sigma_P. A \text{ is root of } T\}$ , then  $S = M(P)$ .*

Hence any element in the model has an idealized proof and anything that has an idealized proof is in the model. This formulation of the declarative semantics in terms of idealized proofs will be used in soundness and completeness proofs in order to distinguish between the case when a query has a finite derivation, from the case when there are only infinite derivations of the query.

**Operational Semantics:** This section defines the operational semantics for co-LP. This requires some infinite tree theory. However, this section only states a few definitions and theorems without proof. The details of infinite tree theory can be found in [4].

The operational semantics given for co-LP is defined as an interleaving of SLD [7] and co-SLD [9]. Where SLD uses sets of syntactic atoms and syntactic term substitutions for states, co-SLD uses finite trees of syntactic atoms along with systems of equations. Of course, the traditional goals of SLD can be extracted from these trees, as the goal of a forest is simply the set of leaves of the forest. Furthermore, where SLD only allows program clauses as state transition rules, co-SLD also allows a special coinductive hypothesis rule for proving coinductive atoms [9]. The coinductive hypothesis rule allows us to compute the greatest fixed-point. Intuitively, it states that if, during execution, a call  $C$  is encountered that unifies with an ancestor call  $A$  that has been seen before, then  $C$  is deleted from the resolvent and the substitution resulting from unification of  $C$  and  $A$  imported into the resolvent. Thus, given the clause:  $p \text{ :- } p$  the query  $! ?- p$  succeeds by the coinductive hypothesis rule (indeed  $\{p\}$  is the gfp of this program). Likewise, given the clause:  $p([1|T]) \text{ :- } p(T)$  the query  $! ?- p(X)$  succeeds by the coinductive hypothesis rule with solution  $X = [1 | X]$  (note that indeed  $\{p([1, \dots])\}$ , where  $[1, \dots]$  denotes an infinite list of 1's, is the gfp of the program).

**Definition 13.** *A system of equations  $E$  is a term unification problem where each equation is of the form  $X = t$ , s.t.  $X$  is a variable and  $t$  a syntactic term.*

**Theorem 2.** (Courcelle) *Every system of equations has a rational mgu.*

**Theorem 3.** (Courcelle) *For every rational substitution  $\sigma$  with domain  $V$ , there is a system of equations  $E$ , such that the most general unifier  $\sigma'$  of  $E$  is equal to  $\sigma$  when restricted to domain  $V$ .*

Without loss of generality, the previous two theorems allow for a solution to a term unification problem to be simultaneously a substitution as well as a system of equations. Given a substitution specified as a system of equations  $E$ , and a term  $A$ , the term  $E(A)$  denotes the result of applying  $E$  to  $A$ .

Now the operational semantics can be defined. The semantics implicitly defines a state transition system. Systems of equations are used to model the part of the state involving unification. The current state of the pending goals is modeled using a forest of finite trees of atoms, as it is necessary to be able to recognize infinite proofs, for coinductive queries. However, an implementation that executes goals in the current resolvent from left to right (as in standard LP), only needs a single stack.

**Definition 14.** *A state  $S$  is a pair  $(F, E)$ , where  $F$  is a finite multi-set of finite trees (syntactic atoms), and  $E$  is a system of equations.*

**Definition 15.** *A transition rule  $R$  of a co-logic program  $P$  is an instance of a clause in  $P$ , with variables consistently renamed for freshness, or  $R$  is a coinductive hypothesis rule of the form  $\nu(\pi, \pi')$ , where  $\pi$  and  $\pi'$  are paths, such that  $\pi$  is a proper prefix of  $\pi'$ .*

Before we can define how a transition rule affects a state, we must define how a tree in a state is modified when an atom is proved to be true. This is called the unmemo function, and it removes memo'ed atoms that are no longer necessary. Starting at a leaf of a tree, the unmemo function removes the leaf and the maximum number of its ancestors, such that the result is still a tree. This involves iteratively removing ancestor nodes of the leaf until an ancestor is reached, which still has other children, and so removing any more ancestors would cause the result to no longer be a tree, as children would be orphaned. When all nodes in a tree are removed, the tree itself is removed.

**Definition 16.** *The unmemo function  $\delta$  takes a tree of atoms  $T$ , a path  $\pi$  in the domain of  $T$ , and returns a forest. Let  $\rho(T, \pi \cdot i)$  be the partial function equal to  $T$ , except that it is undefined at  $\pi \cdot i$ .  $\delta(T, \pi)$  is defined as follows:*

$$\begin{aligned} \delta(T, \pi) &= \{T\} && , \text{ if } \pi \text{ has children in } T \\ \delta(T, \epsilon) &= \emptyset && , \text{ if } \epsilon \text{ is a leaf in } T \\ \delta(T, \pi) &= \delta(\rho(T, \pi), \pi') && , \text{ if } \pi = \pi' \cdot i \text{ is a leaf in } T \end{aligned}$$

The intuitive explanation of the following definition is that (1) a state can be transformed by applying the coinductive hypothesis rule  $\nu(\pi, \pi')$ , whenever in some tree,  $\pi$  is a proper ancestor of  $\pi'$ , such that the two atoms unify. Also, (2) a state can be transformed by applying an instance of a definite clause from the program. In either case, when a subgoal has been proved true, the forest is pruned so as to remove unneeded memos. Also, note that the body of an inductive clause is overwritten on top of the leaf of a tree, as an inductive call need not be memo'ed, since the coinductive hypothesis rule can never be invoked on a memo'ed inductive predicate. When the leaf of the tree is also the root, this

causes the old tree to be replaced with, one or more singleton trees. Coinductive subgoals, on the other hand, need to be memo'ed, in the form of a forest, so that infinite proofs can be recognized.

The state transition system may be nondeterministic, depending on the program, that is, it is possible for states to have more than one outgoing transition as the following definition shows (implementations typically use backtracking to realize nondeterministic execution). We write  $S - x$  to denote the multi-set obtained by removing an occurrence of  $x$  from  $S$ .

**Definition 17.** *Let  $T \in F$ . A state  $(F, E)$  transitions to another state  $((F - T) \cup F', E')$  by transition rule  $R$  of program  $P$  whenever:*

1.  $R$  is an instance of the coinductive hypothesis rule of the form  $\nu(\pi, \pi')$ ,  $p$  is a coinductive predicate,  $\pi$  is a proper prefix of  $\pi'$ , which is a leaf in  $T$ ,  $T(\pi) = p(t'_1, \dots, t'_n)$ ,  $T(\pi') = p(t_1, \dots, t_n)$ ,  $E'$  is the most general unifier for  $\{t_1 = t'_1, \dots, t_n = t'_n\} \cup E$ , and  $F' = \delta(T, \pi')$ .
2.  $R$  is a definite clause of the form  $p(t'_1, \dots, t'_n) \leftarrow B_1, \dots, B_m$ ,  $\pi$  is a leaf in  $T$ ,  $T(\pi) = p(t_1, \dots, t_n)$ ,  $E'$  is the most general unifier for  $\{t_1 = t'_1, \dots, t_n = t'_n\} \cup E$ , and the set of trees of atoms  $F'$  is obtained from  $T$  acc. to the following case analysis of  $m$  and  $p$ :
  - (a) Case  $m = 0$ :  $F' = \delta(T, \pi)$ .
  - (b) Case  $m > 0$  and  $p$  is coinductive:  $F' = \{T'\}$  where  $T'$  is equal to  $T$  except at  $\pi \cdot i$ , &  $T'(\pi \cdot i) = B_i$ , for  $1 \leq i \leq m$ .
  - (c) Case  $m > 0$  and  $p$  is inductive: If  $\pi = \epsilon$  then  $F' = \{node(B_i) \mid 1 \leq i \leq m\}$ . Otherwise,  $\pi = \pi' \cdot j$  for some positive integer  $j$ . Let  $T''$  be equal to  $T$  except at  $\pi' \cdot k$  for all  $k$ , where  $T''$  is undefined. Finally,  $F' = \{T'\}$  where  $T' = T''$  except at  $\pi' \cdot i$ , where  $T'(\pi' \cdot i) = B_i$ , for  $1 \leq i \leq m$ , and  $T'(\pi' \cdot (m + k)) = T(\pi' \cdot k)$ , for  $k \neq j$ .

**Definition 18.** *A transition sequence in program  $P$  consists of a sequence of states  $S_1, S_2, \dots$  and a sequence of transition rules  $R_1, R_2, \dots$ , such that  $S_i$  transitions to  $S_{i+1}$  by rule  $R_i$  of program  $P$ .*

A transition sequence is an execution trace. Execution halts when it reaches a terminal state: all atoms have been proved or a dead-end reached.

**Definition 19.** *The following are two distinguished terminal states:*

1. An accepting state is a state of the form  $(\emptyset, E)$ , where  $\emptyset =$  empty set.
2. A failure state is a non-accepting state with no outgoing transitions.

Finally we can define the execution of a query as a transition sequence through the state transition system induced by the input program, with the start state consisting of the initial query.

**Definition 20.** *A derivation of a state  $(F, E)$  in program  $P$  is a state transition sequence with the first state equal to  $(F, E)$ . A derivation is successful if it ends in an accepting state, and a derivation has failed if it reaches a failure state. We say that a list of syntactic atoms  $A_1, \dots, A_n$ , also called a goal or query, has a derivation in  $P$ , if  $(\{node(A_i) \mid 1 \leq i \leq n\}, \emptyset)$  has a derivation in  $P$ .*

**Equivalence:** Correctness is proved by equating operational & declarative semantics via soundness and completeness theorems. Completeness is restricted to atoms with rational proofs.

**Theorem 4.** (*soundness*) *If the query  $A_1, \dots, A_n$  has a successful derivation in program  $P$ , then  $E(A_1, \dots, A_n)$  is true in program  $P$ , where  $E$  is the resulting variable bindings for the derivation.*

**Theorem 5.** (*completeness*) *Let  $A_1, \dots, A_n \in M(P)$ , s.t. each  $A_i$  has a rational idealized proof; the query  $A_1, \dots, A_n$  has a successful derivation in  $P$ .*

## 5 Conclusions and Future Work

In this paper we presented a comprehensive theory of co-LP and demonstrated its practical applications. Note that most past efforts [3,6,5] do not support coinduction in its most general form and therefore are deficient. A prototype implementation of co-LP has been developed by modifying the YAP Prolog system [8]. Current work also includes extending co-LP's operational semantics with alternating OLDT and co-SLD, so that the operational behavior of inductive predicates can be made to more closely match their declarative semantics. Current work also involves extending the operational semantics of co-LP to allow for finite derivations in the presence of irrational terms and proofs, that is, infinite terms and proofs that do not have finitely many distinct subtrees.

**Acknowledgments.** We are grateful to Vítor Santos Costa and Ricardo Rocha for help with YAP, and Srividya Kona for comments.

## References

1. Krzysztof, R.: Apt. Logic programming. In: Handbook of Theoretical Computer Science, pp. 493–574. MIT Press, Cambridge (1990)
2. Barwise, J., Moss, L.: Vicious Circles: On the Mathematics of Non-Wellfounded Phenomena. CSLI Pub. (1996)
3. Colmerauer, A.: Equations and inequations on finite and infinite trees. In: FGCS'84
4. Courcelle, B.: Fundamental properties of infinite trees. TCS, pp. 95–212 (1983)
5. Hanus, M.: Integration of functions into LP. J. Logic Prog. 19, 20, 583–628 (1994)
6. Jaffar, J., Stuckey, P.J.: Semantics of infinite tree LP. TCS 46(2–3), 141–158 (1986)
7. Lloyd, J.W.: Foundations of LP, 2nd edn. Springer, Heidelberg (1987)
8. Rocha, R., et al.: Theory and Practice of Logic Programming 5(1-2), 161–205, (2005)Tabling Engine That Can Exploit Or-Parallelism. In: ICLP 2001, pp. 43–58 (2001)
9. Simon, L., Mallya, A., Bansal, A., Gupta, G.: Coinductive Logic Programming. In: Etalle, S., Truszczyński, M. (eds.) ICLP 2006. LNCS, vol. 4079, pp. 330–345. Springer, Heidelberg (2006)
10. Simon, L., Mallya, A., Bansal, A., Gupta, G.: Co-Logic Programming: Extending Logic Programming with Coinduction. TR #UTDCS-21-06, UT Dallas (2006)

# Offline/Online Mixing

Ben Adida<sup>1,\*</sup> and Douglas Wikström<sup>2,\*\*</sup>

Harvard, Center for Research on Computation and Society

ben@eecs.harvard.edu, douglas@wikstrom.net

**Abstract.** We introduce an offline precomputation technique for mix-nets that drastically reduces the amount of online computation needed. Our method can be based on any additively homomorphic cryptosystem and is applicable when the number of senders and the maximal bit-size of messages are relatively small.

## 1 Introduction

Suppose some senders  $S_1, \dots, S_N$ , each with input  $m_i$ , want to compute the sorted list  $(m_{\pi(1)}, \dots, m_{\pi(N)})$  while keeping the permutation  $\pi$  secret. A trusted party can provide this service. First, it collects all messages. Then, it shuffles the inputs according to  $\pi$  and outputs the result. A protocol, i.e. a list of machines  $M_1, \dots, M_k$ , that emulates this service is called a *mix-net*, and the parties  $M_1, \dots, M_k$  are referred to as *mix-servers*. The assumption is that each sender  $S_i$  trusts that a certain fraction of the mix-servers  $M_1, \dots, M_k$  is honest. The notion of a mix-net was introduced by Chaum [9].

There are numerous proposals in the literature for how to construct a secure mix-net, but there are also several attacks. A rigorous definition of security of a mix-net was first given by Abe and Imai [1], though they did not construct a scheme satisfying their construction. Wikström [20] gives the first definition of a universally composable (UC) mix-net, and also the first UC-secure construction. In recent work, Wikström [21] gives a more efficient UC-secure scheme and Wikström and Groth [23] describes an adaptively secure construction.

In this paper we assume that a statically UC-secure mix-net can be constructed, and consider to what extent offline precomputation can be used to reduce the amount of online computation needed during execution.

### 1.1 Previous Work

General techniques, e.g., precomputation of re-encryption factors, fixed base exponentiation, and simultaneous exponentiation [16], can be used to lower the online computational complexity of most mix-nets in the literature. However, for the known constructions, it seems difficult to use these methods to completely

---

\* Work done while at MIT, funded by the Caltech/MIT Voting Technology Project.

\*\* Work done while at ETH Zürich, Department of Computer Science.

remove the large number of exponentiations needed in the proofs of shuffles used to provide security against active attacks.

We are not aware of any previous work on mix-nets using our approach, but it is inspired by the ground-breaking work on homomorphic election schemes introduced by Cohen<sup>1</sup> and Fischer [10] and further developed in a long line of papers [5,11,15].

In recent work [3], we consider a related precomputation technique with connections to public key obfuscation. By comparison, the solution we present here requires an individual key for each sender but is much more efficient. Thus, the two solutions are complementary.

## 1.2 Our Contributions

We describe a novel precomputation technique for mix-nets based on additively homomorphic cryptosystems such as the Paillier [19] cryptosystem. Although our technique is universally applicable, it only reduces the online complexity in terms of computation and communication when the number of senders and the maximal bit-size of their messages are reasonably small. We also introduce the notion of concatenation-friendly cryptosystems as a separate tool and prove that such schemes can be constructed from any additively homomorphic cryptosystem. Our technique may be of great value in some practical applications where online computational power is a scarce resource and the result is needed quickly.

## 1.3 Notation

We denote the natural numbers by  $\mathbb{N}$ , the integers by  $\mathbb{Z}$ , the integers modulo  $n$  by  $\mathbb{Z}_n$ , the multiplicative group modulo  $n$  by  $\mathbb{Z}_n^*$ , and the subgroup of squares modulo  $n$  by  $SQ_n$ . We interpret strings as integers in base two when convenient. We write  $a\|b$  to denote the concatenation of the two strings  $a$  and  $b$ . We use  $\text{PT}$  and  $\text{PT}^*$  to denote the set of polynomial time and non-uniform polynomial time Turing machines respectively, and let  $\kappa$  be the main security parameter. We say that a function  $\epsilon(\kappa)$  is negligible if for every constant  $c$  and sufficiently large  $\kappa$  it holds that  $\epsilon(\kappa) < \kappa^{-c}$ . We denote by  $\text{Sort}$  the algorithm that, on input a list of strings, outputs the same strings in lexicographical order. If  $pk$  is the public key of a cryptosystem, we denote by  $M_{pk}$ ,  $C_{pk}$ , and  $R_{pk}$  the plaintext space, the ciphertext space, and the randomness space respectively. We state our results using the Universal Composability (UC) framework [8]. We use slightly non-standard notation in that we use an explicit communication model, denoted  $\mathcal{C}_{\mathcal{I}}$ , that acts as a router between the parties. We refer the reader to [8,22] for details on this variant of the UC model.

## 2 Additively Homomorphic Cryptosystems

There are several homomorphic cryptosystems in the literature, but not all are *additively* homomorphic. For our new scheme, we do not require the

<sup>1</sup> In his later work, Cohen published under the name Benaloh.

cryptosystem to have efficient decryption for all encrypted messages. More precisely, we use the following definitions.

**Definition 1.** A weak cryptosystem  $\mathcal{CS} = (\text{Kg}, \text{E}, \text{D})$  is a cryptosystem except we do not require that  $\text{D}$  run in polynomial time. If there exists polynomial  $T(\cdot)$  and  $\kappa_s(\kappa) > 0$  such that  $\{0, 1\}^{\kappa_s} \subset M_{pk}$  and such that  $\text{D}_{sk}(\text{E}_{pk}(m))$  outputs  $m$  in time  $T(\kappa)$  for every  $(pk, sk) = \text{Kg}(1^\kappa)$  and  $m \in \{0, 1\}^{\kappa_s}$ , we call  $\mathcal{CS}$  a  $\kappa_s$ -cryptosystem.

**Definition 2.** A weak cryptosystem  $\mathcal{CS}$  is homomorphic if for every  $(pk, sk) = \text{Kg}(1^\kappa)$ :

1. The message space  $M_{pk}$  and the randomizer space  $R_{pk}$  are additive abelian groups, and the ciphertext space  $C_{pk}$  is a multiplicative abelian group, and the group operations can be computed in polynomial time given  $pk$ .
2. For every  $m, m' \in M_{pk}$  and  $r, r' \in R_{pk}$ :  $\text{E}_{pk}(m, r)\text{E}_{pk}(m', r') = \text{E}_{pk}(m + m', r + r')$ .

**Definition 3.** A weak homomorphic cryptosystem  $\mathcal{CS}$  is said to be additive if, for every  $(pk, sk) = \text{Kg}(1^\kappa)$  the message space  $M_{pk}$  is the additive modular group  $\mathbb{Z}_n$  for some integer  $n > 1$ . In this case we identify the elements of  $\mathbb{Z}_n$  with their bit-string representations as integers in base two.

*Efficient Examples.* The Paillier cryptosystem [19,12] is additively homomorphic, since  $M_{pk} = \mathbb{Z}_n$ ,  $R_{pk} = \mathbb{Z}_n^*$ , and  $C_{pk} = \mathbb{Z}_{n^2}^*$ , where  $n$  is the  $\kappa$ -bit modulus contained in the public key  $pk$ . Similarly, the Okamoto-Uchiyama cryptosystem [18], a precursor of the Paillier cryptosystem, is additively homomorphic, since  $M_{pk} = \mathbb{Z}_p$ ,  $R_{pk} = \mathbb{Z}_n$ , and  $C_{pk} = \mathbb{Z}_n^*$ , where  $n$  is the  $\kappa$ -bit modulus contained in the public key  $pk$ .

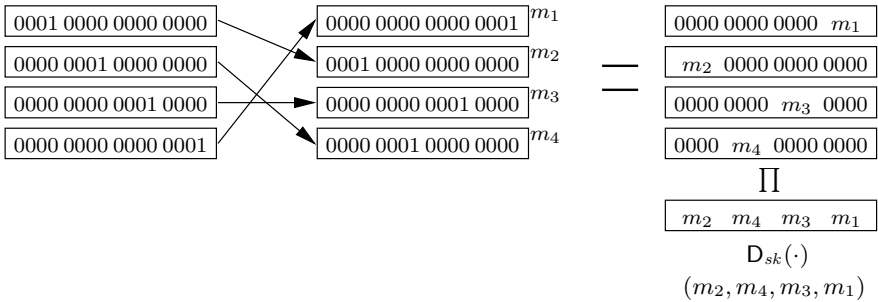
*Inefficient Examples.* The Goldwasser-Micali cryptosystem [14], when based on quadratic residues, is additively homomorphic, since  $M_{pk} = \mathbb{Z}_2$ ,  $R_{pk} = SQ_n$ , and  $C_{pk}$  is the subset of  $\mathbb{Z}_n^*$  with Jacobi symbol 1. This example may be interesting despite its inefficiency, since the quadratic residuosity assumption is considered a very weak assumption. The Boneh-Goh-Nissim cryptosystem [6] can be viewed as an additively homomorphic  $O(\kappa)$ -cryptosystem. This is both inefficient and based on a very strong assumption, but it may still be interesting in connection with our ideas due to its special algebraic properties.

### 3 The Basic Idea

Our construction is simple provided that we use an additive homomorphic  $\kappa_s$ -cryptosystem such that  $N\kappa_m < \kappa_s$ , where  $N$  is the maximal number of senders and  $\kappa_m$  is the maximal bit-size of submitted messages.

The idea can be described as follows. Define  $B_i = 2^{(i-1)\kappa_m}$  for  $i = 1, \dots, N$ . The offline phase produces ciphertexts for the sequence of indexed positions

where the inputs will end up, namely  $B_1, \dots, B_N$ . Then, still in the offline phase, these ciphertexts are re-randomized and shuffled. Each sender is assigned one such encrypted index to use as his effective public key. The sender uses the additive homomorphic property of the cryptosystem to exponentiate his encrypted index to his plaintext value  $m_i$ , thereby creating a ciphertext of the value  $m_i$  offset to that sender's bit position (which remains hidden from the sender). The resulting ciphertext is then sent to the bulletin board. When all inputs are submitted, the offline phase ends. Then, they are aggregated using homomorphic addition. The plaintext of the resulting single ciphertext is the concatenation of all submitted messages, with each message at its appropriate offset. The idea is illustrated in Figure 1



**Fig. 1.** The trivial ciphertexts are shuffled to produce a new list of re-encrypted and permuted ciphertexts. Then each sender uses its assigned ciphertext as a public key and the result is a new list of ciphertexts, where the messages of the senders are embedded. Finally, the mix-servers take the product of the ciphertexts and decrypt a single ciphertext to find the input messages, but in random order.

In the remainder of the paper we relax the restriction  $N\kappa_m \leq \kappa_s$ , give a more detailed description, and prove the security of the scheme, but, before we do so we give a more detailed description of the simple case. In the offline-phase, the mix-servers first form the list of trivial encryptions

$$(C_1, \dots, C_N) = (E_{pk}(B_1, 0), \dots, E_{pk}(B_N, 0)) .$$

Then, they mix the above list to produce a randomly re-encrypted and permuted list of ciphertext on the form

$$(C'_1, \dots, C'_N) = (E_{pk}(B_{\pi(1)}, s_1), \dots, E_{pk}(B_{\pi(N)}, s_N)) .$$

The sender  $S_i$  is then assigned the public key  $pk_i = C'_i$ . To send a message  $m_i \in \{0, 1\}^{\kappa_m}$ , the sender  $S_i$  chooses  $r_i \in R_{pk}$  randomly, computes the ciphertext

$$c_i = pk_i^{m_i} E_{pk}(0, r_i)$$

and writes it on the bulletin board. It also proves knowledge of  $r_i \in R_{pk}$  and  $m_i \in \{0, 1\}^{\kappa_m}$  such that the above holds. When the submission phase is over, the mix-servers compute the product  $c = \prod_{i=1}^N c_i$ . Note that we have



$$\begin{aligned}
 c &= \prod_{i=1}^N pk_i^{m_i} E_{pk}(0, r_i) = E_{pk}(0, r) \prod_{i=1}^N E_{pk}(B_{\pi(i)}, s_i)^{m_i} = E_{pk}\left(\sum_{i=1}^N B_{\pi(i)} m_i, r'\right) \\
 &= E_{pk}\left(\sum_{i=1}^N B^{\pi(i)-1} m_i, r'\right) = E_{pk}(m_{\pi^{-1}(1)} \parallel \dots \parallel m_{\pi^{-1}(N)}, r') ,
 \end{aligned}$$

for  $r = \sum_{i=1}^N r_i$  and  $r' = r + \sum_{i=1}^N s_i m_i$ , since  $m_i \in \{0, 1\}^{\kappa_m}$ . The mix-servers jointly compute  $m'_1 \parallel \dots \parallel m'_N = D_{sk}(c)$ , and output  $\text{Sort}(m'_1, \dots, m'_N)$ .

*The Relation With Homomorphic Election Schemes.* Recall that the idea behind the homomorphic election schemes [10] mentioned in the introduction is to use an additive homomorphic  $\kappa_s$ -cryptosystem and let a sender  $S_i$  encode a vote for party  $j$  by a ciphertext  $c_i = E_{pk}(M^j)$ , where  $M$  is an integer larger than the number of senders  $N$ . The point is that the plaintext of the ciphertext product  $\prod_{i=1}^N c_i$  is of the form  $\sum_{j=0}^{C-1} a_j M^j$ , where  $a_j$  is the number of senders that voted for candidate number  $j$ . If  $C$  is the number of candidates, this approach requires that  $C \log N \leq \kappa_s$ , but one can increase the number of candidates by using several ciphertexts. In some sense, our approach follows by switching the roles played by candidates and senders.

## 4 Model and Definitions

We define some ideal functionalities and the notion of concatenation-friendly cryptosystems to allow us to state our results more easily.

### 4.1 The Ideal Bulletin Board

We assume the existence of an ideal authenticated bulletin board. Each party can write to the bulletin board, nobody can erase anything from the bulletin board, and the messages that appear on the bulletin board are indexed in the order they appear (see the full version [4] for a formal definition).

### 4.2 The Ideal Mix-Net

We use an ideal mix-net functionality similar to the one in [20]. The only essential difference is that we explicitly allow the adversary to prohibit senders from submitting an input. This makes the ideal functionality more realistic.

**Functionality 1 (Mix-Net).** The ideal functionality for a mix-net,  $\mathcal{F}_{MN}$ , running with mix-servers  $M_1, \dots, M_k$ , senders  $S_1, \dots, S_N$ , and ideal adversary  $\mathcal{S}$  proceeds as follows

1. Initialize a list  $L = \emptyset$ , a database  $D$ , a counter  $c = 0$ , and set  $J_S = \emptyset$  and  $J_M = \emptyset$ .

2. Repeatedly wait for inputs

- Upon receipt of  $(S_i, \text{Send}, m_i)$  with  $m_i \in \{0, 1\}^{\kappa_m}$  and  $i \notin J_S$  from  $\mathcal{C}_{\mathcal{I}}$ , store this tuple in  $D$  under the index  $c$ , set  $c \leftarrow c + 1$ , and hand  $(\mathcal{S}, S_i, \text{Input}, c)$  to  $\mathcal{C}_{\mathcal{I}}$ .
- Upon receipt of  $(M_j, \text{Run})$  from  $\mathcal{C}_{\mathcal{I}}$ , store  $(M_j, \text{Run})$  in  $D$  under the index  $c$ , set  $c \leftarrow c + 1$ , and hand  $(\mathcal{S}, M_j, \text{Input}, c)$  to  $\mathcal{C}_{\mathcal{I}}$ .
- Upon receipt of  $(\mathcal{S}, \text{AcceptInput}, c)$  such that something is stored under the index  $c$  in  $D$  do
  - (a) If  $(S_i, \text{Send}, m_i)$  with  $i \notin J_S$  is stored under  $c$ , then append  $m_i$  to  $L$ , set  $J_S \leftarrow J_S \cup \{i\}$ , and hand  $(\mathcal{S}, S_i, \text{Send})$  to  $\mathcal{C}_{\mathcal{I}}$ .
  - (b) If  $(M_j, \text{Run})$  is stored under  $c$ , then set  $J_M \leftarrow J_M \cup \{j\}$ . If  $|J_M| > k/2$ , then sort the list  $L$  lexicographically to form a list  $L'$ , hand  $((\mathcal{S}, M_j, \text{Output}, L'), \{(M_l, \text{Output}, L')\}_{l=1}^k)$  to  $\mathcal{C}_{\mathcal{I}}$  and ignore further messages. Otherwise, hand  $\mathcal{C}_{\mathcal{I}}$  the list  $(\mathcal{S}, M_j, \text{Run})$ .

4.3 The Ideal Mixer

Since our focus in this paper is to minimize the online work needed by the mix-servers and not how to construct a secure mix-net from scratch, we assume the existence of a powerful ideal functionality that allows us to invoke the different phases of a mix-net without going into details. We use this functionality during the offline phase only. Although it is essentially equivalent to an ideal mix-net, we call it a mixer to distinguish it from the ideal mix-net above, and we parameterize it by a cryptosystem. The functionality outputs a public key, waits for a list of ciphertexts to mix, and then finally waits for ciphertexts to decrypt.

**Functionality 2 ( $\mathcal{CS}$ -Mixer).** The ideal functionality for a  $\mathcal{CS}$ -mixer,  $\mathcal{F}_{\text{mixer}}$ , running with mix-servers  $M_1, \dots, M_k$ , senders  $S_1, \dots, S_N$ , and ideal adversary  $\mathcal{S}$  proceeds as follows

1. Set  $J_M = \emptyset$ , compute  $(pk, sk) = \text{Kg}(1^\kappa)$ , and hand  $((\mathcal{S}, \text{PublicKey}, pk), \{(M_j, \text{PublicKey}, pk)\}_{j=1}^k)$  to  $\mathcal{C}_{\mathcal{I}}$ .
2. Wait for an input on the form  $(M_j, \text{Mix}, L_j)$  with  $j \notin J_M$  and set  $J_M \leftarrow J_M \cup \{j\}$ .
  - (a) If there is an  $L = (c_i)_{i=1}^N$  such that  $L_j = L$  for more than  $k/2$  distinct  $j$ , where  $c_i \in C_{pk}$ , choose  $r_i \in R_{pk}$  randomly and compute  $L' = (c_{\pi(1)}E_{pk}(0, r_1), \dots, c_{\pi(N)}E_{pk}(0, r_N))$  for a random  $\pi \in \Sigma_N$ . Then hand  $((\mathcal{S}, \text{Mixed}, L'), \{(M_j, \text{Mixed}, L')\}_{j=1}^k)$  to  $\mathcal{C}_{\mathcal{I}}$ , and go to the next step.
  - (b) Otherwise hand  $(\mathcal{S}, M_j, \text{Mix}, L_j)$  to  $\mathcal{C}_{\mathcal{I}}$  and wait for another input.
3. Repeatedly wait for messages. Upon receiving  $(M_j, \text{Decrypt}, c)$  check if  $c$  has been received. If so set  $J_c \leftarrow J_c \cup \{j\}$ . Otherwise initialize  $J_c = \emptyset$ . If  $|J_c| > k/2$ , then hand  $((\mathcal{S}, \text{Decrypted}, c, D_{sk}(c)), \{(M_j, \text{Decrypted}, c, D_{sk}(c))\}_{i=1}^k)$  to  $\mathcal{C}_{\mathcal{I}}$ , and otherwise hand  $(\mathcal{S}, M_j, \text{Decrypt}, c)$  to  $\mathcal{C}_{\mathcal{I}}$ .

Proving that this functionality can be realized in an efficient and UC-secure way is beyond the scope of this paper. It can be achieved following [21,23].

#### 4.4 Ideal Zero-Knowledge Proof of Knowledge of Plaintexts

We assume the existence of an ideal zero-knowledge proof of knowledge for correct encryption. The corresponding relation is defined below.

**Definition 4 (Plaintext Knowledge).** *Define the relation  $\mathcal{R}_{\text{enc}}$  as consisting of the pairs  $((pk, pk', c), (m, r))$  such that  $c = (pk')^m \mathbf{E}_{pk}(0, r)$  and  $m \in \{0, 1\}^{\kappa_m}$ .*

**Functionality 3 (Zero-Knowledge Proof of Knowledge).** Let  $\mathcal{L}$  be a language given by a binary relation  $R$ . The ideal *zero-knowledge proof of knowledge* functionality  $\mathcal{F}_{\text{ZK}}^R$  of a witness  $w$  to an element  $x \in \mathcal{L}$ , running with provers  $S_1, \dots, S_N$ , and verifiers  $M_1, \dots, M_k$ , proceeds as follows.

1. Upon receipt of  $(S_i, \text{Prover}, x, w)$  from  $\mathcal{C}_{\mathcal{I}}$ , store  $w$  under the tag  $(S_i, x)$ , and hand  $(\mathcal{S}, S_i, \text{Prover}, x, R(x, w))$  to  $\mathcal{C}_{\mathcal{I}}$ . Ignore further messages from  $S_i$ .
2. Upon receipt of  $(M_j, \text{Question}, S_i, x)$  from  $\mathcal{C}_{\mathcal{I}}$ , if  $J_{S_i, x}$  is not initialized set  $J_{S_i, x} = \emptyset$  and otherwise  $J_{S_i, x} \leftarrow J_{S_i, x} \cup \{j\}$ . Let  $w$  be the string stored under the tag  $(S_i, x)$  (the empty string if nothing is stored). If  $|J_{S_i, x}| = k$ , then hand  $((\mathcal{S}, M_j, \text{Verifier}, S_i, x, R(x, w)), \{(M_j, \text{Verifier}, S_i, x, R(x, w))\}_{j=1}^k)$  to  $\mathcal{C}_{\mathcal{I}}$  and otherwise  $(\mathcal{S}, M_j, \text{Question}, S_i, x)$ .

Note that the functionality synchronizes the response. For cryptosystems such as Paillier [19] and ElGamal [13] with messages encoded in the exponent, the above functionality can be efficiently realized using the Naor and Yung [17] double ciphertext trick and an efficient proof of membership in an interval [7].

#### 4.5 Concatenation Friendly Cryptosystems

To simplify the exposition, we introduce the notion of concatenation-friendly cryptosystems. Informally, a concatenation-friendly cryptosystem allows concatenation of plaintexts under the cover of encryption. We show that this feature can be obtained from any additively homomorphic  $\kappa_s$ -cryptosystem for an arbitrary  $\kappa_s > 0$ .

**Definition 5.** *Let  $\mathcal{CS} = (\text{Kg}, \mathbf{E}, \mathbf{D})$  be a  $N\kappa_s$ -cryptosystem. We say that  $\mathcal{CS}$  is  $(N, \kappa_m)$ -concatenation friendly if there exists  $\text{Shift}, \text{Exp} \in \text{PT}$ , such that for every  $\kappa \in \mathbb{N}$  and every  $(pk, sk) = \text{Kg}(1^\kappa)$ :*

1. For every  $m \in \{0, 1\}^{\kappa_m}$  we have  $\mathbf{D}_{sk}(\text{Exp}_{pk}(\mathbf{E}_{pk}(0), m)) = 0$ .
2. For every  $1 \leq t \leq N$  and  $m_c \in \{0, 1\}^{\kappa_m}$ :

$$\mathbf{D}_{sk}(\text{Exp}_{pk}(\mathbf{E}_{pk}(\text{Shift}_{pk}(t)), m_c)) = 0^{(t-1)\kappa_m} \| m_c \| 0^{(N-t)\kappa_m} .$$

3. For every  $m_l \in \{0, 1\}^{(t-1)\kappa_m}$ ,  $m_c \in \{0, 1\}^{\kappa_m}$ ,  $m_r \in \{0, 1\}^{(N-t)\kappa_m}$ :

$$\mathbf{D}_{sk}(\mathbf{E}_{pk}(m_l \| 0^{\kappa_m} \| m_r) \mathbf{E}_{pk}(0^{(t-1)\kappa_m} \| m_c \| 0^{(N-t)\kappa_m})) = m_l \| m_c \| m_r .$$

We abuse notation and write  $c^m$  instead of  $\text{Exp}_{pk}(c, m)$ , and also drop the subscript  $pk$  from  $\text{Shift}_{pk}(\cdot)$ . We stress that, in general, the operation computed by  $\text{Exp}$  is *not* the standard exponentiation operator.

**Proposition 1.** *Let  $N$ ,  $\kappa_m$ , and  $\kappa_s > 0$  be polynomially bounded. If there exists a polynomially indistinguishable additively homomorphic  $\kappa_s$ -cryptosystem, then there exists a  $(N, \kappa_m)$ -concatenation friendly and polynomially indistinguishable  $N\kappa_m$ -cryptosystem.*

*Proof.* Let  $\text{CS}^{\text{ah}} = (\text{Kg}^{\text{ah}}, \text{E}^{\text{ah}}, \text{D}^{\text{ah}})$  be a polynomially indistinguishable additively homomorphic  $\kappa_s$ -cryptosystem for some polynomial  $\kappa_s(\kappa) > 0$ . Define  $\text{Kg}$  equal to  $\text{Kg}^{\text{ah}}$ .

The idea is to “pack” the bits of a message into a list of ciphertexts in such a way that we can “concatenate” messages from  $\{0, 1\}^{\kappa_m}$  under encryption as required by the definition. We assume that an integer  $0 < t_m \leq \kappa_s$  has been fixed and define  $t_r = \lceil \kappa_m / t_m \rceil$ . The integer  $t_r$  decides into how many pieces we divide a message  $m \in \{0, 1\}^{\kappa_m}$ , and  $t_m$  decides how many bits we have in each such piece. Note that we may choose a value of  $t_m$  lower than strictly necessary, so that, later, we can optimize the number of bits encrypted under  $\text{CS}^{\text{ah}}$  depending on the specific values of  $N$ ,  $\kappa_m$ , and  $\kappa_s$ , without breaking the symmetry required for concatenation under the cover of encryption.

On input  $pk$  and  $m \in \{0, 1\}^{N\kappa_m}$ , the encryption algorithm  $\text{E}$  first writes  $m = m_1 \parallel \dots \parallel m_N$  with  $m_j \in \{0, 1\}^{\kappa_m}$ . Then it writes  $m_j = m_{1,j} \parallel \dots \parallel m_{t_r,j}$  with  $m_{i,j} \in \{0, 1\}^{t_m}$ . This gives a  $t_r \times N$ -matrix  $m = (m_{i,j})$ , where the  $j$ th column corresponds to  $m_j$ . Then it defines

$$M_{i,j} = m_{i,jt_M+1} \parallel \dots \parallel m_{i,jt_M+t_M}$$

for  $j = 0, \dots, t'_M$  where  $t_M$  is chosen maximal under the restriction  $t_m t_M \leq \kappa_s$ , and  $t'_M = N/t_M - 1$ . Finally, the algorithm chooses  $r_{i,j} \in R_{pk}^{\text{ah}}$  randomly and defines

$$c = \left( \text{E}_{pk}^{\text{ah}}(M_{i,j}, r_{i,j}) \right)_{i=1, j=0}^{t_r, t'_M} .$$

The decryption algorithm  $\text{D}$  takes as input a secret key  $sk$  and a ciphertext  $c = (c_{i,j})$  and proceeds as follows. It first computes

$$(M_{i,j}) = (\text{D}_{sk}^{\text{ah}}(c_{i,j}))$$

for  $i = 1, \dots, t_r$ ,  $j = 0, \dots, t'_M$  and interprets  $M_{i,j}$  as  $m_{i,jt_M+1} \parallel \dots \parallel m_{i,jt_M+t_M}$  by truncating the string in the natural way. Then, it outputs the concatenation  $m$  of the columns in the matrix  $m = (m_{i,l})$ , where  $i$  ranges over  $\{1, \dots, t_r\}$  and  $l$  ranges over  $\{1, \dots, N\}$ .

The encryption and decryption algorithms obviously run in polynomial time, since each individual operation does, and it is easy to see that an encrypted message is always recovered. Thus,  $\text{CS} = (\text{Kg}, \text{E}, \text{D})$  is a  $N\kappa_m$ -cryptosystem.

The polynomial indistinguishability of the scheme follows by a standard hybrid argument, since a ciphertext essentially consists of a polynomial length list of ciphertexts of a polynomially indistinguishable cryptosystem [14].

It remains to show that the scheme is  $(N, \kappa_m)$ -concatenation friendly. We define multiplication component-wise, i.e.,  $cc' = (c_{i,j})(c'_{i,j}) = (c_{i,j}c'_{i,j})$ . The output of  $\text{Shift}(t)$  is defined as the concatenation of the columns in the  $t_r \times N$ -matrix

$(z_{i,l})$  where  $z_{i,l} = 0$  for all elements except that  $z_{1,t} = z_{2,t} = \dots = z_{t_r,t} = 1$ . In other words the  $t$ th column consists of ones and all other elements are zero. Finally, we define the Exp algorithm as follows. We write  $m = (m_1, \dots, m_{t_r})$  with  $m_i \in \{0, 1\}^{t_m}$  as above. Then we define

$$c^m = (c_{i,j}^{m_i}) .$$

Consider now  $t$  and  $m_c$  as in Definition 5 and denote by  $z = (z_{i,l}) = \text{Shift}(t)$ , and define  $Z_{i,j} = z_{i,jt_M+1} \parallel \dots \parallel z_{i,jt_M+t_M}$  for  $j = 0, \dots, t'_M$ . We have

$$\begin{aligned} E_{pk}(\text{Shift}(t))^m &= ((E_{pk}^{\text{ah}}(Z_{i,j}))_{i=1,j=0}^{t_r,t'_M})^m = (E_{pk}^{\text{ah}}(Z_{i,j})^{m_i})_{i=1,j=0}^{t_r,t'_M} \\ &= (E_{pk}^{\text{ah}}(z_{i,jt_M+1} \parallel \dots \parallel z_{i,jt_M+t_M})^{m_i})_{i=1,j=0}^{t_r,t'_M} . \end{aligned}$$

If we write  $E_{pk}^{\text{ah}}(z_{i,jt_M+1} \parallel \dots \parallel z_{i,jt_M+t_M})^{m_i} = E_{pk}^{\text{ah}}(z'_{i,jt_M+1} \parallel \dots \parallel z'_{i,jt_M+t_M})$ , we may conclude that  $z'_{i,l} = 0$  for all  $i$  and  $l$  except that  $z'_{i,t} = m_i$  for  $i = 1, \dots, t_r$ . In other words, the second requirement is satisfied. Note that if  $\text{Shift}(t)$  is replaced by 0 above, we see in a similar way that the first requirement is satisfied.

Consider now  $m_l$ ,  $m_c$ , and  $m_r$  as in Definition 5 and write  $m = m_l \parallel 0^{\kappa_m} \parallel m_r$  and  $m' = 0^{(t-1)\kappa_m} \parallel m_c \parallel 0^{(N-t)\kappa_m}$ . We have

$$\begin{aligned} E_{pk}(m)E_{pk}(m') &= (E_{pk}^{\text{ah}}(M_{i,j}))_{i=1,j=0}^{t_r,t'_M} (E_{pk}^{\text{ah}}(M'_{i,j}))_{i=1,j=0}^{t_r,t'_M} \\ &= (E_{pk}(m_{i,jt_M+1} \parallel \dots \parallel m_{i,jt_M+t_M}) E_{pk}(m'_{i,jt_M+1} \parallel \dots \parallel m'_{i,jt_M+t_M}))_{i=1,j=0}^{t_r,t'_M} . \end{aligned}$$

From the additive homomorphism of  $\text{CS}^{\text{ah}}$  we conclude that

$$\begin{aligned} E_{pk}(m_{i,jt_M+1} \parallel \dots \parallel m_{i,jt_M+t_M}) E_{pk}(m'_{i,jt_M+1} \parallel \dots \parallel m'_{i,jt_M+t_M}) \\ = E_{pk}(\bar{m}_{i,jt_M+1} \parallel \dots \parallel \bar{m}_{i,jt_M+t_M}) \end{aligned}$$

with  $\bar{m}_{i,l} = m_{i,l}$  for  $l \neq t$  and  $\bar{m}_{i,l} = m'_{i,l}$  otherwise. Thus, the third requirement is satisfied.

Finally, note that it is an easy task to optimize the value of  $t_m$  with regards to minimizing the number of individual ciphertexts.

## 5 Detailed Protocol and Security Analysis

We are now ready to describe the details of our scheme and prove its security.

**Protocol 1 (Online/Offline Mix-Net).** The online/offline mix-net  $\pi_{\text{MN}}^{o/o}$  executing with senders  $S_1, \dots, S_N$ , mix-servers  $M_1, \dots, M_k$ , and ideal adversary  $\mathcal{S}$  proceeds as follows.

SENDER  $S_i$

1. Wait until  $(M_j, \text{SenderPublicKeys}, (pk_i)_{i=1}^N)$  appears on  $\mathcal{F}_{\text{BB}}$  for more than  $k/2$  distinct  $j$ .

2. Wait for an input (Send,  $m_i$ ) with  $m_i \in \{0, 1\}^{\kappa_m}$ . Then choose  $r_i \in R_{pk}$  randomly and compute  $c_i = pk_i^{m_i} E_{pk}(0, r_i)$ .
3. Hand (Prover,  $(pk, pk_i, c_i), (m_i, r_i)$ ) to  $\mathcal{F}_{ZK}^{\mathcal{R}^{enc}}$ .
4. Hand (Send,  $c_i$ ) to  $\mathcal{F}_{BB}$ .

MIX-SERVER  $M_j$

*Offline Phase*

1. Wait for a message (PublicKey,  $pk$ ) from  $\mathcal{F}_{mixer}$ .
2. Form the list  $L = (E_{pk}(\text{Shift}(1), 0), \dots, E_{pk}(\text{Shift}(N), 0))$ . Hand (Mix,  $L$ ) to  $\mathcal{F}_{mixer}$ , and wait until it returns (Mixed,  $(pk_i)_{i=1}^N$ ).
3. Hand (Write, SenderPublicKeys,  $(pk_i)_{i=1}^N$ ) to  $\mathcal{F}_{BB}$ .
4. Initialize  $J_M = \emptyset$  and repeatedly wait for new inputs or the next new message on  $\mathcal{F}_{BB}$ .
  - On input (Run), hand (Write, Run) to  $\mathcal{F}_{BB}$ .
  - If  $(M_j, \text{Run})$  appears on  $\mathcal{F}_{BB}$ , then set  $J_M \leftarrow J_M \cup \{j\}$ . If  $|J_M| > k/2$ , go to Step 5.
  - If  $(S_\gamma, \text{Send}, c_\gamma)$  appears on  $\mathcal{F}_{BB}$  for  $\gamma \notin J_S$  then do:
    - (a) Set  $J_S \leftarrow J_S \cup \{\gamma\}$ .
    - (b) Hand (Question,  $S_\gamma, (pk, pk_\gamma, c_\gamma)$ ) to  $\mathcal{F}_{ZK}^{\mathcal{R}^{enc}}$  and wait for a reply (Verifier,  $S_\gamma, (pk, pk_\gamma, c_\gamma), b_\gamma$ ) from  $\mathcal{F}_{ZK}^{\mathcal{R}^{enc}}$ .

*Online Phase*

5. Let  $J'_S \subset J_S$  be the set of  $\gamma$  such that  $b_\gamma = 1$ . Compute  $c = \prod_{\gamma \in J'_S} c_\gamma$ , hand (Decrypt,  $c$ ) to  $\mathcal{F}_{mixer}$ , and wait until a message (Decrypted,  $c, m$ ) is returned by  $\mathcal{F}_{mixer}$ .
6. Write  $m = m_1 \parallel \dots \parallel m_N$ , where  $m_i \in \{0, 1\}^{\kappa_m}$ , set  $m' = (m_1, \dots, m_N)$ , and return (Output, Sort( $m'$ )).

### 5.1 Online Complexity

The complexity of our scheme depends heavily on the application, the cryptosystem used, the number of parties  $N$  and the maximal bit-size  $\kappa_m$  of messages. The setting where our techniques reduce the online complexity the most is when the verification of the submissions can be considered part of the offline phase and  $N\kappa_m \leq O(\kappa)$ . For this case, the online complexity both in terms of computation and communication between the mix-servers is drastically reduced, as illustrated by the following example.

The most natural practical set-up is to use the Paillier cryptosystem [19] with  $N\kappa_m \leq O(\kappa)$ . In this case, the online complexity consists of performing  $O(N)$  multiplications and  $O(1)$  joint decryptions. This can be done using  $O(k)$  exponentiations, with a small hidden constant. The fastest mix-net based on the Paillier cryptosystem requires at least  $\Omega(kN)$  exponentiations with small constants with precomputation. Thus, we get a speed-up on the order of  $N$ .

We have chosen to consider the submission phase as part of the offline phase. If this is not reasonable, then our techniques are still applicable, but they do

not reduce the complexity as much. In the Paillier example, this would give a speedup on the order of  $k$ . We expect most applications with  $N\kappa \leq O(\kappa)$  to be somewhere between these to extremes.

## 5.2 Security Analysis

We denote by  $\mathcal{M}_l$  the set of static adversaries that corrupt at most  $l$  mix-servers and arbitrarily many senders. The following proposition captures the security properties of the protocol.

**Proposition 2.** *Let  $\mathcal{CS}$  be a concatenation-friendly and polynomially indistinguishable cryptosystem. Then  $\pi_{\text{MN}}^{\text{o/o}}$  securely realizes  $\mathcal{F}_{\text{MN}}$  with respect to  $\mathcal{M}_{k/2}$  adversaries in the  $(\mathcal{F}_{\text{BB}}, \mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{enc}}}, \mathcal{F}_{\text{mixer}})$ -hybrid model.*

We refer the reader to the full version [4] for a proof.

## 6 Conclusion

A mix-net allows any polynomial number  $N$  of senders to send any of exponentially many possible messages, i.e., the only restriction is that  $N\kappa_m$  is polynomial in  $\kappa$ , where  $\kappa_m$  is the maximal bit-size of submitted messages.

The homomorphic election schemes may be viewed as a mix-net with the restriction that  $2^{\kappa_m} \log N \leq O(\kappa)$ , i.e., each sender can send one out of very few messages, but there can be many senders. The advantage of this is that homomorphic election schemes are much more efficient than general mix-nets.

In this paper we have considered the dual restriction  $\kappa_m N \leq O(\kappa)$ , i.e., there can be few senders, but each sender can send one out of many messages. We have shown that, in this case also, there exists a solution that is much more efficient than a general mix-net in the online phase.

## References

1. Abe, M., Imai, H.: Flaws in some robust optimistic mix-nets. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 39–50. Springer, Heidelberg (2003)
2. Adida, B., Wikström, D.: How to shuffle in public. Cryptology ePrint Archive, Report 2005/394 (2005), <http://eprint.iacr.org/>
3. Adida, B., Wikström, D.: How to shuffle in public. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 555–574. Springer, Heidelberg (2007) Accepted for publication at Theory of Cryptography Conference 2007 (full version [2])
4. Adida, B., Wikström, D.: Offline/online-mixing. Cryptology ePrint Archive, Report 2007/143 (2007), <http://eprint.iacr.org/>
5. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: 26th ACM Symposium on the Theory of Computing (STOC), pp. 544–553. ACM Press, New York (1994)

6. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–342. Springer, Heidelberg (2005)
7. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001) (Full version at Cryptology ePrint Archive, Report 2000/067, <http://eprint.iacr.org>, October, 2001)
9. Chaum, D.: Untraceable electronic mail, return addresses and digital pseudo-nyms. *Communications of the ACM* 24(2), 84–88 (1981)
10. Cohen, J., Fischer, M.: A robust and verifiable cryptographically secure election scheme. In: 28th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 372–382. IEEE Computer Society Press, Los Alamitos (1985)
11. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
12. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier’s probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
13. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31(4), 469–472 (1985)
14. Goldwasser, S., Micali, S.: Probabilistic encryption. *Journal of Computer and System Sciences* 28(2), 270–299 (1984)
15. Katz, J., Myers, S., Ostrovsky, R.: Cryptographic counters and applications to electronic voting. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 78–92. Springer, Heidelberg (2001)
16. Menezes, A., Oorschot, P., Vanstone, S.: *Handbook of Applied Cryptography*. CRC Press, Boca Raton (1997)
17. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attack. In: 22th ACM Symposium on the Theory of Computing (STOC), pp. 427–437. ACM Press, New York (1990)
18. Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)
19. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
20. Wikström, D.: A universally composable mix-net. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 315–335. Springer, Heidelberg (2004)
21. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle (Full version [22]). In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 273–292. Springer, Heidelberg (2005)
22. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. *Cryptology ePrint Archive, Report 2004/137* (2005) <http://eprint.iacr.org/>
23. Wikström, D., Groth, J.: An adaptively secure mix-net without erasures. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 276–287. Springer, Heidelberg (2006)



# Fully Collusion Resistant Black-Box Traitor Revocable Broadcast Encryption with Short Private Keys

Jun Furukawa<sup>1</sup> and Nuttapong Attrapadung<sup>2</sup>

<sup>1</sup> NEC Corporation, Japan  
j-furukawa@ay.jp.nec.com

<sup>2</sup> Research Center for Information Security,  
National Institute of Advanced Industrial Science and Technology, Japan  
n.attrapadung@aist.go.jp

**Abstract.** Broadcast encryption schemes enable senders to efficiently broadcast ciphertexts to a large set of receivers in a way that only non-revoked receivers can decrypt them. Black-box traitor revocable broadcast encryption schemes are broadcast encryption schemes that enable a tracer, who is given a pirate decoder, to identify traitors by black-box accessing the given pirated decoder and to revoke traitors so identified. In this paper, we propose a fully collusion resistant black-box traitor revocable broadcast encryption scheme in which the size of each private key is constant, the size of the public key is proportional to the number of receivers, and the sizes of ciphertexts are sub-linear with respect to the number of receivers. The encryption procedure in our scheme requires only a public key. The tracing procedure in it requires only a public key and black-box access to a resettable pirate decoder. The security of our scheme is proved in the generic bilinear group model if the subgroup decision assumption holds.

**Keywords:** Black-Box, Traitor Tracing, Revocation, Fully Collusion Resistant.

## 1 Introduction

Broadcast encryption schemes are cryptosystems that enable senders to efficiently broadcast ciphertexts to a large set of receivers such that only non-revoked receivers can decrypt them. Their notion was first introduced by Berkovits in [4] and was given formal analysis by Fiat and Naor in [17]. Many schemes, such as [17, 11, 27, 25, 20, 16, 3, 19, 2, 11], have been proposed since then, and their main purpose is to decrease private key size, ciphertext length, public key size, and computational costs for encryption and decryption. One notable scheme is the one proposed by Boneh et al. in [11].

Black-box traitor revocable broadcast encryption schemes are variants of broadcast encryption schemes that enable a tracer, which is given a pirate decoder, to identify traitors by black-box accessing the pirated decoder and to

revoke traitors so identified. The first such scheme was introduced by D. Naor et al. in [25] as a broadcast encryption scheme with a trace and revoke system. Since the number of receivers in broadcast encryption schemes is potentially large, pirate decoders are always a serious threat, and that is why trace and revoke systems are useful for them. Recently, Boneh et al. proposed in [13] one such a scheme that has many desirable properties.

Black-box traitor tracing schemes are those closely related to black-box traitor revocable broadcast encryption schemes. These are cryptosystems that enable senders to efficiently send ciphertexts to a large number of receivers such that, given a pirate decoder, at least one receiver who has leaked its key can be traced by black-box accessing this pirate decoder. After the notion of such schemes was first introduced by Chor et al. in [10], many schemes, [21,26,31,32,18,9,27,28,25,23,16,15], including non black-box schemes, have been proposed. One notable scheme is that proposed by Boneh et al. in [12].

In this paper, we propose a new black-box traitor revocable broadcast encryption scheme. Its encryption and revocation procedure require only a public key. Black-box traitor revocation is possible even if all receivers are colluding and the scheme is secure as a broadcast encryption scheme even if all the non-revoked receivers are colluding. In this paper, we call such a scheme fully collusion resistant black-box public key traitor revocable public key broadcast encryption scheme. We have proved the security of our scheme under the subgroup decision (SD) assumption in the generic bilinear group model.

Our scheme is a careful combination of a modified version of the broadcast encryption scheme in [11] and the traitor tracing scheme in [12]. The modified broadcast encryption scheme is designed to prevent revoked receivers from being able to check whether ciphertexts have been generated correctly or not. This is achieved by making decrypted results of revoked users to be different each other, which prevents them from comparing their results. Because of this property, during when the tracer iteratively revokes traitors, already revoked traitors are unable to recognized whether it is revoked or not so as to hinder the tracer from tracing other traitors. This modified scheme is then integrated with the scheme in [12] to form the proposed scheme. We note that Boneh et al. used a similar approach in [13] but with a different variant of broadcast encryption scheme, which results to achieve different complexity.

The complexity of our scheme is compared to those of the previous scheme [13] and related schemes [12] and [11]<sup>1</sup> in Table 1. In particular, the size of private keys in the scheme [13] is  $\mathcal{O}(\sqrt{N})$ , larger than our  $\mathcal{O}(1)$ . By way of contrast, the size of public key in their scheme is  $\mathcal{O}(\sqrt{N})$ , shorter than our  $\mathcal{O}(N)$ . These differences come from a difference in the respective newly constructed broadcast encryption portions of the two combination schemes. Boneh's new broadcast encryption scheme in [13] requires longer private keys but requires only shorter public key. The order of the cost for black-box revocation in both schemes is at most  $(\kappa/\epsilon)N^3$  where  $\kappa$  is the security parameter for revocation and  $\epsilon$  is the

---

<sup>1</sup> The case when the size of ciphertext is  $\mathcal{O}(\sqrt{N})$ .

maximum probability that the adversary is allowed to succeed in attacks. This can be improved if we use technique of binary search etc.

The adaptive security of Boneh's scheme can be proven in the standard model, but ours can only be proven in the generic bilinear model. This is the cost we paid to shorten the private key. Nevertheless, the security of assumptions that the scheme in [13] is proven only in generic bilinear group model.

An advantage of our scheme over that in [13] can be found in the following case. Consider a scenario in which each receiver carries his private key in a secure device such as a smart card and decrypts messages with the help of untrusted platform such as a PC connected to the Internet which is flooded with virus and spy-ware. The part of decryption procedures that requires their private keys are done within their smart cards so that there is no chance of their private keys being stolen.

The memory size and computational cost required by our scheme and that in [13] is given also in Table 1. We note that even if the secure devices are handed fake data by untrusted platforms, this secure device works no more than a decryption oracle of our basic scheme.

Although the public key length of our scheme is larger than that in [13], this can never be disadvantage since each receiver requires only  $\mathcal{O}(\sqrt{N})$  size part of this public key. If the public key is locally stored in the decoder in our scheme, the size of this stored data is no more than that of stored private key in the decoder of [13]. Moreover, this portion of public key in our scheme can be stored in the untrusted platform which are supposed to have rather large storage, while private keys cannot be so without risk in the scheme of [13].

**Table 1.** Comparison of schemes

|                                     | Our scheme              | [13]                    | [12]                    | [11]             |
|-------------------------------------|-------------------------|-------------------------|-------------------------|------------------|
| private key length                  | $\mathcal{O}(1)$        | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(1)$        | $\mathcal{O}(1)$ |
| ciphertext length                   | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(1)$ |
| public key length                   | $\mathcal{O}(N)$        | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(N)$ |
| public key length for each receiver | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(N)$ |
| black-box traitor revoking          | public-key              | public-key              | N/A                     | N/A              |
| Memory size of secure device        | $\mathcal{O}(1)$        | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(1)$        | $\mathcal{O}(1)$ |
| Computational cost of secure device | $\mathcal{O}(1)$        | $\mathcal{O}(\sqrt{N})$ | $\mathcal{O}(1)$        | $\mathcal{O}(1)$ |

## Organization

The paper is organized as follows: Section 2 specifies the model used and defines the security requirements for a black-box traitor revocable broadcast encryption scheme. Section 3 reviews a number of basic facts and complexity assumptions that our scheme relies on. Section 4 presents our basic scheme and describes how we can obtain from the basic scheme a scheme that is fully secure.

## 2 The Model and Security Requirements

### 2.1 The Model

Four types of players participate in black-box public-key traitor revocable broadcast encryption schemes: a membership authority  $\mathcal{M}$ , traitors  $\mathcal{T}$ , encrypters  $\mathcal{E}$ , and receivers. We let the total number of receivers be  $N$  and  $i$ -th receiver be  $\mathcal{R}_i$ .


Each scheme consists of five algorithms: Gen, Ext, Enc, Dec, and Revoke.

**Gen:** A probabilistic setup algorithm that, given the number of receivers  $N$ , outputs a public key  $pkey$  and a registration key  $mkey$ .  $(pkey, mkey) \leftarrow \text{Gen}(N)$ .  $mkey$  is given to  $\mathcal{M}$ . We assume that  $pkey$  includes  $N$ .

**Ext:** A probabilistic algorithm for  $\mathcal{M}$  that, given  $mkey$ ,  $pkey$ , and an index  $i$  of receiver, outputs a private key  $skey_i$  for  $i$ -th receiver.  $skey_i \leftarrow \text{Ext}(pkey, mkey, i)$ .

**Enc:** A probabilistic algorithm for encrypters that, given a receiver set  $S \subseteq \{1, \dots, N\}$  and a public key  $pkey$ , outputs a shared key  $key$  and a header  $hdr$ .  $(key, hdr) \leftarrow \text{Enc}(S, pkey)$ .

**Dec:** A probabilistic algorithm for receivers that, given a  $hdr$  for  $S$ ,  $i \in S$ ,  $skey_i$ , and the public key  $pkey$ , outputs  $key$ .  $key \leftarrow \text{Dec}(S, i, skey_i, hdr, pkey)$ .

**Revoke:** A probabilistic algorithm for  $\mathcal{T}$  that, given a receiver set  $S^\dagger$ ,  $pkey$ , a black-box access to a pirate decoder  $\mathcal{D}$ , and a probability parameter  $\epsilon$ , outputs an honest receiver set  $S^\ddagger \subset S^\dagger$ .  $S^\ddagger \leftarrow \text{Revoke}^{\mathcal{D}}(S^\dagger, pkey, \epsilon)$  

### 2.2 Security Requirements

Let us now consider formal definitions of security requirements for black-box public key traitor revocable public key broadcast encryption schemes. The **correctness** is defined as usual and is omitted in this version.

We next consider requirements for the scheme to be a secure broadcast encryption and to be able to revoke traitors in black-box manner until the pirate decoder is no longer able to decrypt ciphertexts but all the honest and legitimate decoders are still able to decrypt them. Since we here require the scheme to be a broadcast encryption that is key indistinguishable against adversaries, it is natural to require the scheme to be also key indistinguishable against pirates, so do we require it. Therefore, our requirements are stronger than those in [13] and are balanced ones. We do not need to consider revocation queries of adversaries and pirate decoders since Revoke requires no private data.

**Definition 1 (Key Distinguishing Game).** Suppose  $N$  is given to  $\mathcal{C}$  and  $\mathcal{A}$ .

1. **Key Generation Phase:**  $\mathcal{C}$  runs  $(pkey, mkey) \leftarrow \text{Gen}(N)$  and provides  $pkey$  to  $\mathcal{A}$ .  $\mathcal{C}$  keeps  $pkey$  and  $mkey$ .  $\mathcal{C}$  keeps a colluder set  $T$  which is initially set to be  $\emptyset$ .
2. **Query Phase:**  $\mathcal{A}$  adaptively sends the following queries:
  - **Extraction:**  $\mathcal{A}$  sends  $\mathcal{C}$  an index  $i \in \{1, \dots, N\}$ . Then,  $\mathcal{C}$  runs  $skey_i \leftarrow \text{Ext}(pkey, mkey, i)$  and provides  $skey_i$  to  $\mathcal{A}$ .  $\mathcal{C}$  replace  $T$  with  $T \cup i$ .

<sup>2</sup> Note that Revoke requires only public data.

- **Decryption:**  $\mathcal{A}$  sends  $\mathcal{C}$  a tuple  $(S \subset \{1, \dots, N\}, i \in S, \text{hdr})$ . Then,  $\mathcal{C}$  runs  $(\text{key}, \text{hdr}) \leftarrow \text{Dec}(S, i, \text{skey}_i, \text{hdr}, \text{pkey})$  and provides  $\text{key}$  to  $\mathcal{A}$ . While sending the above two types of queries,  $\mathcal{A}$  sends  $\mathcal{C}$  the following **challenge query** once and is provided a challenge by  $\mathcal{C}$  in turn.
3. **Challenge Query:**  $\mathcal{A}$  sends  $\mathcal{C}$  a receiver set  $S^* \subset \{1, \dots, N\}$ . Then,  $\mathcal{C}$  runs  $(\text{key}^*, \text{hdr}^*) \leftarrow \text{Enc}(S^*, \text{pkey})$ , randomly chooses  $b \in_R \{0, 1\}$  and  $\text{key}_{1-b}^* \in_R \mathcal{K}$ , and assigns  $\text{key}_b^* = \text{key}^*$ .  $\mathcal{C}$  finally sends to  $\mathcal{A}$   $(\text{hdr}^*, \text{key}_0^*, \text{key}_1^*)$ . Here,  $\mathcal{C}$  only allows  $S^*$  such that  $S^* = \{1, \dots, N\} \setminus T^*$ , where  $T^*$  be the colluder set  $T$  at the end of the game.
4. **Answer Phase:** At the end of game,  $\mathcal{A}$  sends  $b' \in \{0, 1\}$ . Then  $\mathcal{C}$  outputs 1 if  $b' = b$ , otherwise outputs 0.

Let  $\text{AdvKI}(\mathcal{A}, N)$  denote the probability that  $\mathcal{C}$  outputs 1 in this game.

**Definition 2 (Key Indistinguishable).** We say a traitor revocable broadcast encryption scheme is  $(t, \epsilon, N, q_d)$ -key indistinguishable if, for all  $t$ -times algorithm  $\mathcal{A}$  who makes a total of  $q_d$  decryption queries and at most  $N$  extraction queries, we have that

$$|\text{AdvKI}(\mathcal{A}, N) - 1/2| \leq \epsilon.$$

**Definition 3 (Pirate Distinguisher Game).** Suppose  $\mathcal{C}$  and  $\mathcal{A}$  are given  $N$ .

1. **Key Generation Phase:** This phase is exactly the same as that in Definition 1.  $\mathcal{C}$  keeps a colluder set  $T$  which is initially set to be  $\emptyset$ .
2. **Query Phase:** In this phase,  $\mathcal{A}$  adaptively sends extraction queries and decryption queries as in the query phase in Definition 1.  $\mathcal{C}$  updates  $T$  as in Definition 1.
3. **Challenge Query:**  $\mathcal{A}$  sends  $\mathcal{C}$  a pirate decoder  $\mathcal{D}$  and  $S^\dagger \in \{1, \dots, N\}$ , where  $\mathcal{D}$  is a probabilistic circuit that takes, as input, a receiver set  $S$ , a header  $\text{hdr}$ ,  $\text{pkey}$ , and two candidates of keys  $\text{key}_0$  and  $\text{key}_1$ , outputs  $b'$ . We let  $T^*$  denote the colluder set  $T$  when the challenge query is sent.<sup>3</sup>
4. **Revocation and Answer Phase:**  $\mathcal{C}$  runs  $S^\ddagger \leftarrow \text{Revoke}^{\mathcal{D}}(S^\dagger, \text{pkey}, \epsilon)$ . Next,  $\mathcal{C}$  runs  $(\text{key}^*, \text{hdr}^*) \leftarrow \text{Enc}(S^\ddagger, \text{pkey})$ , randomly chooses  $b \in_R \{0, 1\}$  and  $\text{key}_{1-b}^* \in_R \mathcal{K}$ , and assigns  $\text{key}_b^* = \text{key}^*$ .  $\mathcal{C}$  gives  $\mathcal{D}$   $(\text{hdr}^*, \text{key}_0^*, \text{key}_1^*)$ . Then,  $\mathcal{D}$  returns  $b' \in \{0, 1\}$ . Finally,  $\mathcal{C}$  outputs 1 if  $b' = b$  and  $S^\dagger \setminus S^\ddagger \subset T^*$ ; otherwise 0.

Let  $\text{AdvPD}(\mathcal{A}, N)$  denote the probability that  $\mathcal{C}$  outputs 1 in this game.

**Definition 4 (Black-Box Traitor Revocable).** We say a traitor revocable broadcast encryption scheme is  $(t, \epsilon, N, q_d, t_R)$ -black-box traitor revocable if, for

<sup>3</sup> If the pirate decoder is one which outputs key when it is given  $S^\dagger$ ,  $\text{hdr}$ , and  $\text{pkey}$ , then we construct another pirate decoder that distinguishes keys. This can be done by using the original pirate decoder as a black-box. Given  $S^\dagger$ ,  $\text{hdr}$ ,  $\text{pkey}$ ,  $\text{key}_0$ ,  $\text{key}_1$ , gives the new pirate decoder  $S^\dagger$ ,  $\text{hdr}$ ,  $\text{pkey}$  to the original pirate decoder and obtains the output  $\text{key}^*$ . If  $\text{key}^* = \text{key}_{b'}$ , it outputs  $b'$ ; otherwise outputs random bit.

all  $t$ -times algorithm  $\mathcal{A}$  who makes a total of  $q_d$  decryption queries and at most  $N$  extraction queries before sending  $\mathcal{D}$ , we have  $t_R$ -time revocation algorithm Revoke such that

$$|\text{AdvPD}(\mathcal{A}, N) - 1/2| \leq \epsilon.$$

### Limitation and Extension of the Model

It is natural to consider key indistinguishability of pirate decoder (it may be called as pirate distinguisher) since we require that our scheme is key indistinguishable as a broadcast encryption scheme, which is a strictly stronger requirement than key recoverability. Although it is also natural by the same reason to consider pirate decoders that can ask decryption queries and extraction queries after the challenge ciphertext is given to it, we do not consider such pirate decoders. This is because no tracer is able to answer such queries of  $\mathcal{D}$  as long as it is given only public key. We consider public-key revocability is nice property that it worth to achieve it by giving up to consider such stronger pirate decoders.

It is more natural to consider indistinguishability of ciphertexts rather than indistinguishability of session keys. However, we adopted the latter for simplicity. Since ciphertexts in broadcast encryption scheme tend to be long, it is better to construct the entire scheme in hybrid manner. Hence, considering only its header, i.e., key encapsulation mechanism, for concrete construction is practical. We note that there is no difficulty in constructing such a scheme that ciphertexts are indistinguishable and that traitor revocation is possible from pirate decoder who distinguishes ciphertexts, from our proposed scheme and a symmetric key cryptosystem. Thus, the simplification that we choose loose no generality.

The goal of revocation in our model is to output receiver such  $\mathcal{S}^\ddagger$  that the corresponding pirate decoder is no longer able to decrypt ciphertexts for this receiver set  $\mathcal{S}^\ddagger$ . However, pirate decoder may be able to decode ciphertext for another receiver set  $\mathcal{S} \neq \mathcal{S}^\ddagger$  even if  $\mathcal{S} \subsetneq \mathcal{S}^\ddagger$ . That means the tracers in this model are required to check whether or not any of previously found pirate decoders return to be effective every time when they use a new receiver set. This problem cannot be solved within black-box revocation paradigm.

## 3 Preliminaries

We describe basic facts and assumptions that our scheme depends on.

### Definition 5 Bilinear groups of composite order

1.  $\mathcal{G}$  and  $\mathcal{G}_T$  are two cyclic groups of composite order  $n = pq$  where  $p$  and  $q$  are distinct primes.
2.  $e : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G}_T$  is an efficient map such that:
  - (i) *Bilinear*: for all  $u, v \in \mathcal{G}$  and  $\alpha, \beta \in \mathbb{Z}/n\mathbb{Z}$ , we have  $e(u^\alpha, v^\beta) = e(u, v)^{\alpha\beta}$ .
  - (ii) *Non-degenerate*: there exists  $g \in \mathcal{G}$  such that  $e(g, g)$  has order  $n$  in  $\mathcal{G}_T$ .

We let  $\hat{\mathcal{G}}$  and  $\hat{\mathcal{G}}_T$  be, respectively, order  $q$  subgroups of  $\mathcal{G}$  and  $\mathcal{G}_T$ ,  $\check{\mathcal{G}}$  and  $\check{\mathcal{G}}_T$  be, respectively, order  $p$  subgroups of  $\mathcal{G}$  and  $\mathcal{G}_T$ . Let  $g, \hat{g}, \check{g}$  be, respectively, generators of  $\mathcal{G}, \hat{\mathcal{G}},$  and  $\check{\mathcal{G}}$ .

The generic group model [29] for the bilinear group of prime order is defined in [5]. Note that, although our scheme works in bilinear group of **composite** order, we use the generic group model of bilinear group of **prime** order rather than that of composite order to prove its security. The property for which the composite order is essential is proven under the SD assumption.

**The generic group of prime order:** Let us consider the case the bilinear groups of prime order  $p$  are  $\check{\mathcal{G}}$  and  $\check{\mathcal{G}}_T$  and  $\check{g}$  is a generator of  $\check{\mathcal{G}}$ . In this model, elements of  $\check{\mathcal{G}}$  and  $\check{\mathcal{G}}_T$  appear to be encoded as unique random strings, so that no property other than equality can be directly tested by th adversary. There exist three oracles in this model. Those are, oracles that perform group operations in each  $\check{\mathcal{G}}$  and  $\check{\mathcal{G}}_T$  and an oracle that performs paring  $e$ . The opaque encoding of an element in  $\check{\mathcal{G}}$  is modeled as an injective function  $\psi : \mathbb{Z}/p\mathbb{Z} \rightarrow \Sigma \subset \{0, 1\}^*$ , which maps all  $\alpha \in \mathbb{Z}/p\mathbb{Z}$  to the string representation  $\psi(\check{g}^\alpha)$  of  $\check{g}^\alpha \in \check{\mathcal{G}}$ . We similarly define  $\psi_T : \mathbb{Z}/p\mathbb{Z} \rightarrow \Sigma_T$  for  $\check{\mathcal{G}}_T$ . The attacker communicates with the oracles using the  $\psi$ -representations of the group elements only.

**Definition 6 (SD Assumption).** *The Subgroup Decision (SD) problem is stated as follows. Given  $\mathcal{G}, \hat{\mathcal{G}}, g,$  and  $\hat{g}$  as defined before where the order of  $\hat{\mathcal{G}}$  is unknown, decide whether an element  $s$  is a random member of the subgroup  $\hat{\mathcal{G}}$  or a random element of the full group  $\mathcal{G}$ . We say that an algorithm  $\mathcal{A}$  has an advantage  $\epsilon_{SD}$  in solving the SD Problem if*

$$\left| \Pr[\mathcal{A}(n, g, \hat{g}, s) = 1 : s \in_R \hat{\mathcal{G}}] - \Pr[\mathcal{A}(n, g, \hat{g}, s) = 1 : s \in_R \mathcal{G}] \right| \geq \epsilon_{SD}.$$

*We say the  $(t, \epsilon_{SD})$ -subgroup decision assumption holds if no  $t$  time adversary has an advantage  $\epsilon_{SD}$  in solving the subgroup decision problem.*

### 4 Basic Scheme and Full Scheme

We now propose, as our basic scheme, a public key traitor revocable public-key broadcast encryption scheme that is  $(t, \epsilon, N, 0)$ -key indistinguishable and  $(t, \epsilon, N, 0, t_R)$ -black-box traitor revocable under the subgroup decision assumption in the generic bilinear group model. At the end of this section, we briefly present how our full schemes can be constructed.

The key point of our basic scheme lies in the new broadcast encryption portion. Assume that the reader is familiar with the broadcast encryption in [11], where we will now borrow some notations to describe intuition for our modification. We can say that, in the original scheme proposed in [11], the result of decryption by all non-revoked receivers is the data of the form

$$K = e(g_i, C_1)/e(d_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, C_0) = e(g, g)^{\alpha^{\ell+1}}.$$

On the other hand, the result of decryption by every revoked receiver is

$$e(g_i, C_1)/e(d_i \cdot \prod_{j \in S} g_{n+1-j+i}, C_0) = e(g, g)^0 = 1.$$

From this result, all the revoked receivers are able to check whether the ciphertext is correctly generated or not by checking whether or not the result of its decryption coincides with 1. If such a broadcast encryption is used to generate traitor revocable broadcast encryption, already revoked receivers are able to check whether a given ciphertext is generated by Probe in Construction **11** or not. That is the event  $|\hat{p}_i - \hat{p}_{i+1}| \geq \epsilon/2N$  during the execution of Probe can be prevented from occurring even if  $i \notin S^\dagger$ . Hence, the procedure of tracing  $i \in S^\ddagger$  and revoking  $i$  by  $S^\ddagger \rightarrow S^\ddagger \setminus i$  is not effective any more.

**Basic Scheme**

In our scheme, the result of decryption by all non-revoked receivers is the data of the form  $K = e(g, m) \cdot e(g, g)^0$ . However, that of  $(i, j)$ -th revoked receiver is

$$e(g, m) \cdot e(g, g)^{\sigma_j \theta_i \alpha^{\ell+1}}.$$

That values  $\{e(g, g)^{\sigma_j \theta_i \alpha^{\ell+1}}\}_{i,j}$  vary among revoked receivers prevents them from checking the validity of ciphertexts even if they are colluding. Boneh et al. also proposed new broadcast encryption scheme in **13** that solves the same problem and additionally achieve adaptive security at the cost of longer private keys.

Let  $\Lambda$  be the product set  $\{1, \dots, \ell\}^2$  such that each element corresponds to each receiver,  $\Lambda_j \subset \Lambda$  be a subset  $\{(i, j)\}_{i \in \{1, \dots, \ell\}}$ , and let  $p, q$  be sufficiently large primes,  $n = pq$ ,  $\mathcal{G}$  and  $\mathcal{G}_T$  be order  $n$  (elliptic curve) cyclic groups such that an efficient bilinear map  $e : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G}_T$  exists,  $\hat{\mathcal{G}}$  and  $\check{\mathcal{G}}$  be subgroups of  $\mathcal{G}$  such that their orders are  $q$  and  $p$  respectively. Let  $\bar{S}$  be the complement of  $S$  in  $\{1, \dots, \ell\}^2$ . Similar notation is used for other sets such as  $S^*, S^\dagger, S^\ddagger$ .

**Gen:** Given  $\ell$ , Gen chooses a composite order  $n$  cyclic groups  $\mathcal{G}$  and  $\mathcal{G}_T$  with  $e$ . Gen randomly chooses  $(\hat{g}, g) \in_R \hat{\mathcal{G}} \times \mathcal{G}, m \in_R \mathcal{G}, \check{g} \in_R \check{\mathcal{G}}, (\xi, \alpha) \in_R (\mathbb{Z}/n\mathbb{Z})^2, (\beta_j)_{j=1, \dots, \ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell, (\delta_j)_{j=1, \dots, \ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell, (\gamma_i)_{i=1, \dots, \ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell, (\theta_k)_{k=1, \dots, \ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell, (\lambda, \pi) \in_R (\mathbb{Z}/n\mathbb{Z})^2$ .

Next, Gen generates

$$\begin{aligned} (\hat{h}, h) &= (\hat{g}^\xi, g^\xi), \hat{M} = e(\hat{g}, m), M = e(g, m), (v, w) = (g^\pi, \check{g}^\lambda h^\pi) \\ (\hat{b}_j, \hat{h}_j)_{j=1, \dots, \ell} &= (\hat{g}^{\beta_j}, \hat{h}^{\beta_j})_{j=1, \dots, \ell}, (b_j, h_j)_{j=1, \dots, \ell} = (g^{\beta_j}, h^{\beta_j})_{j=1, \dots, \ell} \\ (\hat{g}_i)_{i=1, \dots, 2\ell} &= (\hat{g}^{\alpha^i})_{i=1, \dots, 2\ell}, (g_i)_{i=1, \dots, 2\ell} = (g^{\alpha^i})_{i=1, \dots, 2\ell} \\ (\hat{d}_j)_{j=1, \dots, \ell} &= (\hat{g}^{\delta_j})_{j=1, \dots, \ell}, (d_j)_{j=1, \dots, \ell} = (g^{\delta_j})_{j=1, \dots, \ell} \\ (a_i)_{i=1, \dots, \ell} &= (g^{\gamma_i})_{i=1, \dots, \ell}, (x_{k,i})_{k=1, \dots, \ell; i=1, \dots, 2\ell} = (g_i^{\theta_k})_{k=1, \dots, \ell; i=1, \dots, 2\ell} \end{aligned}$$

Finally, Gen outputs

$$\begin{aligned} pkey &= \{M, (g, h), (b_j, h_j)_{j=1, \dots, \ell}, (g_i)_{i=1, \dots, \ell, \ell+2, \dots, 2\ell}, (d_j)_{j=1, \dots, \ell}, \\ &\quad \hat{M}, (\hat{g}), (\hat{b}_j, \hat{h}_j)_{j=1, \dots, \ell}, (\hat{g}_i)_{i=1, \dots, \ell, \ell+2, \dots, 2\ell}, (\hat{d}_j)_{j=1, \dots, \ell}, \\ &\quad (a_i)_{i=1, \dots, \ell}, (x_{k,i})_{k=1, \dots, \ell; i=1, \dots, \ell, \ell+2, \dots, 2\ell}, (v, w)\} \\ mkey &= \{m, \alpha, (\delta_j, \beta_j)_{j=1, \dots, \ell}, (\gamma_i)_{i=1, \dots, \ell}, (\theta_k)_{k=1, \dots, \ell}\} \end{aligned}$$



Ext: Given a receiver identity  $(i, j) \in \{1, \dots, \ell\}^2$ ,  $mkey$ , and  $pkey$ , Ext generates and output  $skey_{ij} = k_{ij} = mg^{\delta_j \alpha^i \theta_i + \beta_j \gamma_i}$

Enc: Given  $S \subseteq \Lambda$ , Enc randomly chooses  $K \in \mathcal{G}_T$ ,  $(\sigma_j)_{j=1, \dots, \ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell$ ,  $(\epsilon_i)_{i=1, \dots, \ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell$ ,  $(\tau, \rho) \in_R (\mathbb{Z}/n\mathbb{Z})^2$  and generates, for  $i, j = 1, \dots, \ell$ ,

$$(C_j, \hat{e}_j, \hat{f}_j, \hat{r}_j, \hat{t}_j) = \left( K \cdot \hat{M}^{\sigma_j \tau}, \hat{g}^{\sigma_j \tau}, (\hat{d}_j \prod_{k \in \bar{S} \cap \Lambda_j} \hat{g}_{\ell+1-k})^{\sigma_j \tau}, \hat{b}_j^{\sigma_j}, \hat{h}_j^{\rho \sigma_j} \right)$$

$$(s_i, u_i) = (a_i^\tau h^{\rho \epsilon_i}, g^{\epsilon_i}).$$

Then, Enc outputs, a tuple  $hdr_S = ((C_j, \hat{e}_j, \hat{f}_j, \hat{r}_j, \hat{t}_j)_{j=1, \dots, \ell}, (s_i, u_i)_{i=1, \dots, \ell})$ .

Dec: Given  $hdr_S$ ,  $pkey$ , and  $skey_{ij}$  such that  $(i, j) \in S$ , Dec outputs

$$key = K = C_j \cdot \frac{e(x_{i,i}, \hat{f}_j)}{e(\hat{e}_j, k_{ij} \prod_{k \in \bar{S} \cap \Lambda_j} x_{i, \ell+1-k+i})} \cdot \frac{e(\hat{r}_j, s_i)}{e(\hat{t}_j, u_i)}.$$

Revoke: The Revoke, as is given in Construction 2 below, works by repeatedly invoking the following Probe in Construction 1

**Construction 1** (Probe:). Suppose that  $S \subseteq \Lambda$  and  $(\bar{i}, \bar{j}) \in S$ ,  $\kappa$  are given. Probe randomly chooses  $K = e(g, g)^k \in_R \mathcal{G}_T$ ,  $(\sigma_j)_{j=1, \dots, \ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell$ ,  $(\tau, \rho) \in_R \mathbb{Z}/n\mathbb{Z}$ ,  $(K'_j = e(g, g)^{k'_j})_{1, \dots, j=\bar{j}-1} \in_R (\mathcal{G}_T)^{\bar{j}-1}$ ,  $(\sigma'_j)_{1, \dots, j=\bar{j}-1} \in_R (\mathbb{Z}/n\mathbb{Z})^{\bar{j}-1}$  and generates  $(C_j, \hat{e}_j, \hat{f}_j, \hat{r}_j, \hat{t}_j)_{j=1, \dots, \ell}$  as in the following:

For  $j > \bar{j}$ ,

$$\left( K \cdot e(\hat{g}, m)^{\sigma_j \tau}, \hat{g}^{\sigma_j \tau}, (\hat{d}_j \prod_{k \in S \cap \Lambda_j} \hat{g}_{\ell+1-k})^{\sigma_j \tau}, \hat{b}_j^{\sigma_j}, \hat{h}_j^{\rho \sigma_j} \right).$$

For  $j = \bar{j}$  when  $(\bar{i}, \bar{j}) = (1, 1)$ ,

$$\left( K \cdot e(\hat{g}, m)^{\sigma_{\bar{j}} \tau}, \hat{g}^{\sigma_{\bar{j}} \tau}, (\hat{d}_{\bar{j}} \prod_{k \in S \cap \Lambda_{\bar{j}}} \hat{g}_{\ell+1-k})^{\sigma_{\bar{j}} \tau}, \hat{b}_{\bar{j}}^{\sigma_{\bar{j}}}, \hat{h}_{\bar{j}}^{\rho \sigma_{\bar{j}}} \right).$$

For  $j = \bar{j}$  when  $(\bar{i}, \bar{j}) \neq (1, 1)$ ,

$$\left( K \cdot e(g, m)^{\sigma_{\bar{j}} \tau}, g^{\sigma_{\bar{j}} \tau}, (d_{\bar{j}} \prod_{k \in S \cap \Lambda_{\bar{j}}} g_{\ell+1-k})^{\sigma_{\bar{j}} \tau}, b_{\bar{j}}^{\sigma_{\bar{j}}}, h_{\bar{j}}^{\rho \sigma_{\bar{j}}} \right).$$

For  $j < \bar{j}$ ,

$$\left( K'_j \cdot e(g, m)^{\sigma_j \tau}, g^{\sigma_j \tau}, (d_j \prod_{k \in S \cap \Lambda_j} g_{\ell+1-k})^{\sigma_j \tau}, b_j^{\sigma_j}, h_j^{\rho \sigma_j} \right).$$

Next, Probe randomly chooses  $(\epsilon_i)_{i=1, \dots, \ell} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell$ ,  $(\epsilon'_i)_{i=1, \dots, \bar{i}} \in_R (\mathbb{Z}/n\mathbb{Z})^\ell$  and generates  $(s_i, u_i)_{i=1, \dots, \ell}$  as in the following:

$$\text{For } i \geq \bar{i}, (a_i^\tau h^{\rho \epsilon_i}, g^{\epsilon_i}).$$

$$\text{For } i < \bar{i}, (a_i^\tau h^{\rho \epsilon_i} w^{\rho \epsilon'_i}, g^{\epsilon_i} v^{\epsilon'_i}).$$

Then, Enc outputs, a tuple  $\text{hdr}_S = ((C_j, \hat{e}_j, \hat{f}_j, \hat{r}_j, \hat{t}_j)_{j=1, \dots, \ell}, (s_i, u_i)_{i=1, \dots, \ell})$ .

**Construction 2 (Revoke:).** In this construction, the following procedure Estimate() is repeatedly invoked. Procedure  $\hat{p}_i \leftarrow \text{Estimate}(\text{pkey}, S^\ddagger, i, \epsilon, k)$  is as in the following:

1. Do the following  $8(\kappa/\epsilon)N$  times:

(i) Run  $(\text{key}^*, \text{hdr}^*) \leftarrow \text{Probe}(\text{pkey}, S^\ddagger, i)$ , randomly choose  $b \in_{\mathcal{R}} \{0, 1\}$  and  $\text{key}_{1-b}^* \in_{\mathcal{R}} \mathcal{K}$ , and assign  $\text{key}_b^* = \text{key}^*$ . Then, send to  $\mathcal{D}$

$$(\text{hdr}^*, \text{key}_0^*, \text{key}_1^*).$$

(ii)  $\mathcal{D}$  returns  $b' \in \{0, 1\}$ . Then compare whether or not  $b' = b$  holds.

2. Based on the above  $8(\kappa/\epsilon)N$  comparisons, calculate  $\hat{p}_i$  as the estimated probability that  $b = b'$  holds.

Now, Revoke is as in the following:

1. Revoke is given  $\text{pkey}, S^\ddagger$ , and black-box access to a pirate decoder  $\mathcal{D}$ . It sets  $S^\ddagger = S^\ddagger$ .

2. Until Revoke stops, do the following:

(i) Run  $\hat{p}_1 \leftarrow \text{Estimate}(\text{pkey}, S^\ddagger, 1, \epsilon, k)$ .

(ii) For  $i = 2$  to  $N + 1$  do the following:

i. If  $i \notin S^\ddagger$ ,  $\hat{p}_i = \hat{p}_{i-1}$ .

ii. If  $i \in S^\ddagger$ , Revoke runs  $\hat{p}_i \leftarrow \text{Estimate}(\text{pkey}, S^\ddagger, i, \epsilon, k)$

(iii) For all  $i$  such that  $|\hat{p}_i - \hat{p}_{i+1}| \geq \epsilon/2N$ ,  $S^\ddagger \leftarrow S^\ddagger \setminus i$ . If there is no such  $i$ , output  $S^\ddagger$  and stop, otherwise, go to Step 2a.

**Theorem 1.** We assume  $(t, \epsilon_{SD})$ -subgroup decision assumption and  $\epsilon/N < 1/8$ . Then, the basic scheme is  $(t, \epsilon, N, 0, t_R)$ -black-box traitor revocable in the generic bilinear group model. Here

$$\epsilon \leq \frac{16(\sqrt{N} + t + 28)^7}{p} + 8N\epsilon_{SD}$$

$$t_R \leq t + 8(\kappa/\epsilon)N^3\Theta$$

where  $\Theta$  is the time for single run of Probe and several constant number of operations such as coin flipping etc..

**Theorem 2.** We assume the  $(t, \epsilon_{SD})$ -subgroup decision assumption. Then, the basic scheme is  $(t, \epsilon, N, 0)$ -key indistinguishable in the generic bilinear group model. Here

$$\epsilon \leq \frac{16(\sqrt{N} + t + 28)^7}{p} + 8N\epsilon_{SD}.$$

The proofs of the above two theorems are omitted in this version.

## Full Scheme

The basic scheme is secure only  $(t, \epsilon, N, 0)$ -key indistinguishable and  $(t, \epsilon, N, 0, t_R)$ -black-box traitor revocable. That is, it is secure only against chosen plaintext and adaptive key extraction attacks. We want to construct black-box traitor revocable broadcast encryption schemes that are  $(t, \epsilon, N, q_d \neq 0)$ -key indistinguishable and  $(t, \epsilon, N, q_d \neq 0, t_R)$ -black-box traitor revocable, i.e., they are secure against “adaptive chosen ciphertexts” and adaptive key extraction attacks. This is possible if we use the technique of [14] or the technique of [30]. The latter technique requires the existence of random oracles. Since these are trivial extensions of the basic scheme, we omit the detail of their constructions here.

## Generic Group Model and Open Problem

The security of the scheme is proven only in the generic bilinear group model. This is because our scheme has less redundancy to embed a problem. If we have less redundancy than a adaptively secure protocol, we can embed a problem only in the pre-selected challenge. If we have less redundancy than a selectively secure protocol, we can embed only the adversary-strategy-specific-problem in its challenge. Our scheme achieved its efficiency by this tradeoff. The problem whose difficulty that the security of our scheme depends on varies according to the adversaries behavior, difficulty of each can be proven in the generic bilinear group model. We believe we need an essential improvement to escape from this tradeoff. Such an improvement is still an open problem.

## References

1. Anzai, J., Matsuzaki, N., Matsumoto, T.: A Quick Group Key Distribution Scheme with Entity “Revocation”. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 333–347. Springer, Heidelberg (1999)
2. Attrapadung, N., Imai, H.: Graph-Decomposition-Based Frameworks for Subset-Cover Broadcast Encryption and Efficient Instantiations. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 100–120. Springer, Heidelberg (2005)
3. Attrapadung, N., Kobara, K., Imai, H.: Sequential Key Derivation Patterns for Broadcast Encryption and Key Predistribution Schemes. In: Lai, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 374–391. Springer, Heidelberg (2003)
4. Berkovits, S.: How To Broadcast A Secret. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 535–541. Springer, Heidelberg (1991)
5. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
7. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)

8. Boneh, D., Boyen, X., Shacham, H.: Short Group Signature. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
9. Boneh, D., Franklin, M.K.: An Efficient Public Key Traitor Tracing Scheme. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 338–353. Springer, Heidelberg (1999)
10. Chor, B., Fiat, A., Naor, M.: Tracing Traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
11. Boneh, D., Gentry, C., Waters, B.: Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
12. Boneh, D., Sahai, A., Waters, B.: Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
13. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: ACM Conference on Computer and Communications Security 2006, pp. 211–220 (2006)
14. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
15. Chabanne, H., Phan, D.H., Pointcheval, D.: Public Traceability in Traitor Tracing Schemes. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 542–558. Springer, Heidelberg (2005)
16. Dodis, Y., Fazio, N.: Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack. *Public Key Cryptography*, pp. 100–115 (2003)
17. Fiat, A., Naor, M.: Broadcast Encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
18. Fiat, A., Tassa, T.: Dynamic Traitor Training. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 354–371. Springer, Heidelberg (1999)
19. Goodrich, M.T., Sun, J.Z., Tamassia, R.: Efficient Tree-Based Revocation in Groups of Low-State Devices. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 511–527. Springer, Heidelberg (2004)
20. Halevy, D., Shamir, A.: The LSD Broadcast Encryption Scheme. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 47–60. Springer, Heidelberg (2002)
21. Kurosawa, K., Desmedt, Y.: Optimum Traitor Tracing and Asymmetric Schemes. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, Springer, Heidelberg (1998)
22. Miyaji, A., Nakabayashi, M., Takano, S.: Characterization of Elliptic Curve Traces under FR-Reduction. In: Won, D. (ed.) ICISC 2000. LNCS, vol. 2015, pp. 90–108. Springer, Heidelberg (2001)
23. Mitsunari, S., Sakai, R., Kasahara, M.: A new Traitor tracing. *IEICE Trans. Fundamentals E85-A(2)*, 481–484 (2002)
24. Naor, M., Yung, M.: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In: STOC 1990, pp. 427–437 (1990)
25. Naor, D., Naor, M., Lotspiech, J.: Revocation and Tracing Schemes for Stateless Receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
26. Naor, M., Pinkas, B.: Threshold Traitor Tracing. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 502–517. Springer, Heidelberg (1998)
27. Naor, M., Pinkas, B.: Efficient Trace and Revoke Schemes. *Financial Cryptography* pp. 1–20 (2000)

28. Safavi-Naini, R., Wang, Y.: Sequential Traitor Tracing. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 316–332. Springer, Heidelberg (2000)
29. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
30. Shoup, V., Gennaro, R.: Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)
31. Stinson, D.R., Wei, R.: Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes. *SIAM J. Discrete Math.* 11(1), 41–53 (1998)
32. Stinson, D.R., Wei, R.: Key Preassigned Traceability Schemes for Broadcast Encryption. *Selected Areas in Cryptography*, 144–156 (1998)

# Succinct Ordinal Trees Based on Tree Covering

Meng He<sup>1</sup>, J. Ian Munro<sup>1</sup>, and S. Srinivasa Rao<sup>2</sup>

<sup>1</sup> Cheriton School of Computer Science, University of Waterloo, Canada  
`{mhe, imunro}@uwaterloo.ca`

<sup>2</sup> Computational Logic and Algorithms Group, IT University of Copenhagen,  
Denmark  
`ssrao@itu.dk`

**Abstract.** Various methods have been used to represent a tree of  $n$  nodes in essentially the information-theoretic minimum space while supporting various navigational operations in constant time, but different representations usually support different operations. Our main contribution is a succinct representation of ordinal trees, based on that of Geary *et al.* [7], that supports all the navigational operations supported by various succinct tree representations while requiring only  $2n + o(n)$  bits. It also supports efficient level-order traversal, a useful ordering previously supported only with a very limited set of operations [8].

Our second contribution expands on the notion of a single succinct representation supporting more than one traversal ordering, by showing that our method supports two other encoding schemes as abstract data types. In particular, it supports extracting a word ( $O(\lg n)$  bits) of the balanced parenthesis sequence [1] or depth first unary degree sequence [3; 4] in  $O(f(n))$  time, using at most  $n/f(n) + o(n)$  additional bits, for any  $f(n)$  in  $O(\lg n)$  and  $\Omega(1)$ .

## 1 Introduction

Trees are fundamental data structures in computer science. There are essentially two forms. An *ordinal tree* is a rooted tree in which the children of a node are ordered and specified by their rank, while in a *cardinal tree* of degree  $k$ , each child of a node is identified by a unique number from the set  $[k]$ . In this paper, we mainly consider ordinal trees.

The straightforward representation of trees explicitly associates with each node the pointers to its children. Thus, an ordinal tree of  $n$  nodes is represented by  $\Theta(n \lg n)$  bits. This representation allows straightforward, efficient parent-to-child navigation in trees. However, as current applications often consider very large trees, such a representation often occupies too much space.

To solve this problem, various methods have been proposed to encode an ordinal tree of  $n$  nodes in  $2n + o(n)$  bits, which is close to the information-theoretic minimum (as there are  $\binom{2n}{n}/(n+1)$  different ordinal trees), while supporting various navigational operations efficiently. These representations are based on various traversal orders of the nodes in the tree: *preorder* in which each node is visited before its descendants, *postorder* in which each node is visited after

its descendants, and DFUDS *order* in which all the children of a node are visited before its other descendants [3; 4]. However, different representations of trees usually support different sets of navigational operations. A new problem is to design a succinct representation that supports all the navigational operations of various succinct tree structures. We consider the following operations:

- **child**( $x, i$ ): the  $i^{\text{th}}$  child of node  $x$  for  $i \geq 1$ ;
- **child\_rank**( $x$ ): the number of left siblings of node  $x$  plus 1;
- **depth**( $x$ ): the depth of  $x$ , i.e. the number of edges in the rooted path to node  $x$ ;
- **level\_anc**( $x, i$ ): the  $i^{\text{th}}$  ancestor of node  $x$  for  $i \geq 0$  (given a node  $x$  at depth  $d$ , its  $i^{\text{th}}$  ancestor is the ancestor of  $x$  at depth  $d - i$ );
- **nbdesc**( $x$ ): the number of descendants of node  $x$ ;
- **degree**( $x$ ): the degree of node  $x$ , i.e. the number of its children;
- **height**( $x$ ): the height of the subtree rooted at node  $x$ ;
- **LCA**( $x, y$ ): the lowest common ancestor of nodes  $x$  and  $y$ ;
- **distance**( $x, y$ ): the number of edges of the shortest path between nodes  $x$  and  $y$ ;
- **leftmost\_leaf**( $x$ ) (**rightmost\_leaf**( $x$ )): the leftmost (or rightmost) leaf of the subtree rooted at node  $x$ ;
- **leaf\_rank**( $x$ ): the number of leaves before node  $x$  in preorder plus 1;
- **leaf\_select**( $i$ ): the  $i^{\text{th}}$  leaf among all the leaves from left to right;
- **leaf\_size**( $x$ ): the number of leaves of the subtree rooted at node  $x$ ;
- **node\_rank**<sub>PRE/POST/DFUDS</sub>( $x$ ): the position of node  $x$  in the preorder, postorder or DFUDS order traversal of the tree;
- **node\_select**<sub>PRE/POST/DFUDS</sub>( $r$ ): the  $r^{\text{th}}$  node in the preorder, postorder or DFUDS order traversal of the tree;
- **level\_leftmost**( $i$ ) (**level\_rightmost**( $i$ )): the first (or last) node visited in a preorder traversal among all the nodes whose depths are  $i$ ;
- **level\_succ**( $x$ ) (**level\_pred**( $x$ )): the *level successor* (or *predecessor*) of node  $x$ , i.e. the node visited immediately after (or before) node  $x$  in a preorder traversal among all the nodes whose depths are equal to **depth**( $x$ ).

## 1.1 Related Work

Jacobson’s succinct tree representation [8] was based on the *level order unary degree sequence* (LOUDS) of a tree, which lists the nodes in a level-order traversal<sup>1</sup> of the tree and encode their degrees in unary. With this, Jacobson [8] encoded an ordinal tree in  $2n + o(n)$  bits to support the selection of the first child, the next sibling, and the parent of a given node in  $O(\lg n)$  time under the bit probe model. Clark and Munro [6] further showed how to support the above operations in  $O(1)$  time under the word RAM model with  $\Theta(\lg n)$  word size<sup>2</sup>.

As LOUDS only supports a very limited set of operations, various researchers have proposed different ways to represent ordinal trees using  $2n + o(n)$  bits. There are three main approaches based on (see Table 1 for a complete list of operations that each of them supports):

<sup>1</sup> The ordering puts the root first, then all of its children, from left to right, followed by all the nodes at each successive level (depth).

<sup>2</sup> We use  $\lg n$  to denote  $\lceil \log_2 n \rceil$ .

**Table 1.** Navigational operations supported in  $O(1)$  time on succinct ordinal trees using  $2n + o(n)$  bits

| operations  | BP [5, 10–13] | DFUDS [3, 4, 9] | old TC [7] | new TC |
|---|---------------|-----------------|------------|--------|
| child, child_rank   | ✓             | ✓               | ✓          | ✓      |
| depth, level_anc  | ✓             | ✓               | ✓          | ✓      |
| nbdesc, degree  | ✓             | ✓               | ✓          | ✓      |
| node_rank <sub>PRE</sub> , node_select <sub>PRE</sub>     | ✓             | ✓               | ✓          | ✓      |
| node_rank <sub>POST</sub> , node_select <sub>POST</sub>   | ✓             |                 | ✓          | ✓      |
| height  | ✓             |                 |            | ✓      |
| LCA, distance   | ✓             | ✓               |            | ✓      |
| leftmost_leaf, rightmost_leaf                             | ✓             | ✓               |            | ✓      |
| leaf_rank, leaf_select, leaf_size                         | ✓             | ✓               |            | ✓      |
| node_rank <sub>DFUDS</sub> , node_select <sub>DFUDS</sub> |               | ✓               |            | ✓      |
| level_leftmost, level_rightmost                           |               |                 |            | ✓      |
| level_succ, level_pred                                    | ✓             |                 |            | ✓      |

- *Balanced parenthesis sequence* (BP) [11]. The BP sequence of a given tree can be obtained by performing a depth-first traversal, and outputting an opening parenthesis each time a node is visited, and a closing parenthesis immediately after all its descendants are visited.
- *Depth first unary degree sequence* (DFUDS) [3; 4]. The DFUDS sequence represents a node of degree  $d$  by  $d$  opening parentheses followed by a closing parenthesis. All the nodes are listed in preorder (an extra opening parenthesis is added to the beginning of the sequence), and each node is numbered by its opening parenthesis in its parent’s description (DFUDS number).
- *Tree covering* TC [7]. There is an algorithm to cover an ordinal tree with a set of mini-trees, each of which is further covered by a set of micro-trees.

### 1.2 Our Results

Our first result is to extend the succinct ordinal trees based on tree covering by Geary *et al.* [7] to support all the operations on trees proposed in other work:

**Theorem 1.** *An ordinal tree of  $n$  nodes can be represented using  $2n + o(n)$  bits to support all the operations in Section 7 in  $O(1)$  time under the word RAM model with  $\Theta(\lg n)$  word size.*

We compare our results with existing results in Table 1, in which the columns BP and DFUDS list the results of tree representations based on balanced parentheses and DFUDS, respectively, and the columns old TC and new TC list the results by Geary *et al.* [7] and our results, respectively.

Our second result deals with BP and DFUDS representations as abstract data types, showing that any operation to be supported by BP or DFUDS in the future can also be supported by TC efficiently:



**Theorem 2.** *Given a tree represented by TC, any  $O(\lg n)$  consecutive bits of its BP or DFUDS sequence can be computed in  $O(f(n))$  time, using at most  $n/f(n) + O(n \lg \lg n / \lg n)$  extra bits, for any  $f(n)$  where  $f(n) = O(\lg n)$  and  $f(n) = \Omega(1)$  under the word RAM model with  $\Theta(\lg n)$  word size.*

## 2 Preliminaries

### 2.1 Bit Vectors

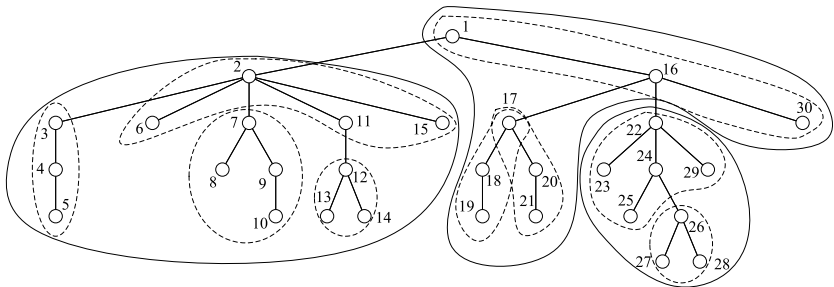
A key structure for many succinct data structures is a bit vector  $B[1..n]$  that supports *rank* and *select* operations. Operations  $\text{rank}_1(B, i)$  and  $\text{rank}_0(B, i)$  return the number of 1s and 0s in  $B[1..i]$ , respectively. Operations  $\text{select}_1(B, i)$  and  $\text{select}_0(B, i)$  return the positions of  $i^{\text{th}}$  1 and 0, respectively. Lemma 1 addresses the problem, in which part (a) is from [6; 8], and part (b) is from [14].

**Lemma 1.** *A bit vector  $B$  of length  $n$  with  $t$  1s can be represented using either: (a)  $n+o(n)$  bits, or (b)  $\lg \binom{n}{t} + O(n \lg \lg n / \lg n)$  bits, to support the access to each bit,  $\text{rank}_1(B, i)$ ,  $\text{rank}_0(B, i)$ ,  $\text{select}_1(B, i)$ , and  $\text{select}_0(B, i)$  in  $O(1)$  time.*

### 2.2 Succinct Ordinal Tree Representation Based on Tree Covering

Geary *et al.* [7] proposed an algorithm to cover an ordinal tree by *mini-trees* (the *tier-1 cover*) of size  $\Theta(M)$  for a given parameter  $M$  (they choose  $M = \lceil \lg^4 n \rceil$ ). Any two mini-trees computed by this algorithm either do not intersect, or only intersect at their common root. They showed that the size of any mini-tree is at most  $3M - 4$ , and the size of any mini-tree that does not contain the root of the tree is at least  $M$ . They further use the same algorithm with the parameter  $M' = \lceil \lg n / 24 \rceil$  to cover each mini-tree by a set of *micro-trees* (the *tier-2 cover*). See Figure 1 for an example, in which all the nodes are numbered in preorder.

Geary *et al.* [7] list the mini-trees  $t_1, t_2, \dots$  in an order such that in a preorder traversal of  $T$ , either the root of  $t_i$  is visited before that of  $t_{i+1}$ , or if these two mini-trees share the same root, the children of the root in  $t_i$  are visited before the



**Fig. 1.** An example of covering an ordinal tree with parameters  $M = 8$  and  $M' = 3$ , in which the solid curves enclose mini-trees and dashed curves enclose micro-trees

children of the root in  $t_{i+1}$ . The  $i^{\text{th}}$  mini-tree in the above sequence is denoted by  $\mu^i$ . All the micro-trees in a mini-tree are also listed in the same order, and the  $j^{\text{th}}$  micro-tree in mini-tree  $\mu^i$  is denoted by  $\mu_j^i$ . When the context is clear, they also refer to this micro-tree using  $\mu_j$ . A node is denoted by its preorder number from an external viewpoint. To further relate a node to its position in its mini-tree and micro-tree, they define the  $\tau$ -name of a node  $x$  to be a triplet  $\tau(x) = \langle \tau_1(x), \tau_2(x), \tau_3(x) \rangle$ , which means that node  $x$  is the  $\tau_3(x)^{\text{th}}$  node visited in a preorder traversal of micro-tree  $\mu_{\tau_2(x)}^{\tau_1(x)}$ . For a node that exists in more than one mini-tree and/or micro-tree, its lexicographically smallest  $\tau$ -name is its *canonical* name, and the corresponding copy of the node is its *canonical* copy.

To enable the efficient retrieval of the children of a given node, Geary *et al.* [7] proposed the notion of *extended micro-trees*. To construct extended micro-trees, each micro-tree is initially made into an extended micro-tree, and each of its nodes is an *original node*. Every node  $x$  that is the root of a micro-tree is promoted into the extended micro-tree to which its parent,  $y$ , belongs. See [7] for the details of promoting a node, and the properties of extended micro-trees. The main data structures by Geary *et al.* [7] are designed to represent each individual extended micro-tree. They take  $2n + o(n)$  bits in total and can be used as building blocks to construct the tree representation at a higher level.

### 3 New Operations Based on Tree Covering (TC)

We now extend the succinct tree representation proposed by Geary *et al.* [7] to support more operations on ordinal trees, by constructing  $o(n)$ -bit auxiliary data structures in addition to their main data structures that use  $2n + o(n)$  bits. As the conversion between the preorder number and the  $\tau$ -name of a given node can be done in constant time [7], we omit the steps of performing such conversions in our algorithms (e.g. we may return the  $\tau$ -name of a node directly when we need to return its preorder number). We use  $T$  to denote the (entire) ordinal tree.

#### 3.1 height

**Definition 1.** Node  $x$  is a **tier-1** (or **tier-2**) **preorder changer** if  $x = 1$ , or if nodes  $x$  and  $(x - 1)$  are in different mini-trees (or micro-trees).

For example, in Figure 1, nodes 1, 2, 16, 22, and 30 are tier-1 preorder changers. Nodes 16, 17, 20, 22, 26 and others are tier-2 preorder changers. It is obvious that all the tier-1 preorder changers are also tier-2 preorder changers. By Lemma 2.2 in [7], each tier-1 (or tier-2) preorder changer can be mapped to a unique tier-1 (or tier-2) boundary node defined by Geary *et al.* [7]. As each mini-tree (or micro-tree) has at most 2 tier-1 (or tier-2) boundary nodes:

**Lemma 2.** The total number of tier-1 (or tier-2) preorder changers is at most twice the number of mini-trees (or micro-trees).

To compute  $\text{height}(x)$ , we compute  $x$ 's number of descendants,  $d$ , using  $\text{nbdesc}$  (from [7], see Section 1). Then all the descendants of  $x$  are nodes  $x + 1, x +$

$2, \dots, x+d$ . We have the formula:  $\text{height}(x) = \max_{i=1}^d (\text{depth}(x+i)) - \text{depth}(x) + 1$ . Therefore, the computation of  $\text{height}(x)$  can be reduced to answering the *range maximum query* (i.e. retrieving the leftmost maximum value among the elements in a given range of the array) on the conceptual array  $D[1..n]$ , where  $D[j] = \text{depth}(j)$  for  $1 \leq j \leq n$ . For any node  $j$ , we can compute  $\text{depth}(j)$  in  $O(1)$  time (7). We now show how to support the range maximum query on  $D$  using  $o(n)$  additional bits without storing  $D$  explicitly.

Based on Bender and Farach-Colton’s algorithm (2), Sadakane (15) showed how to support the range minimum/maximum query in  $O(1)$  time on an array of  $n$  integers using  $o(n)$  additional bits. As we do not explicitly store  $D$ , we cannot use this approach directly (Sadakane (15) combined this approach with an array of integers encoded in the form of balanced parentheses). However, these  $o(n)$ -bits auxiliary data structures provide constant-time support for the range minimum/maximum query without accessing the array, when the starting and ending positions of the range are multiples of  $\lg n$  (i.e. the given range is of the form  $[k \lg n..l \lg n]$ , where  $k$  and  $l$  are integers). We thus construct these  $o(n)$ -bit auxiliary structures to support this special case on the conceptual array  $D$ . To support the general case, we construct (we assume that the  $i^{\text{th}}$  tier-1 and tier-2 preorder changers are numbered  $y_i$  and  $z_i$ , respectively):

- A bit vector  $B_1[1..n]$ , where  $B_1[i] = 1$  iff node  $i$  is a tier-1 preorder changer;
- A bit vector  $B'_1[1..n]$ , where  $B'_1[i] = 1$  iff node  $i$  is a tier-2 preorder changer;
- An array  $C_1[1..l_1]$  ( $l_1$  denotes the number of tier-1 preorder changers), where  $C_1[i] = \tau_1(y_i)$ ;
- An array  $C'_1[1..l'_1]$  ( $l'_1$  denotes the number of tier-2 preorder changers), where  $C'_1[i] = \tau_2(z_i)$ ;
- An array  $E[1..l'_1]$ , where  $E[i]$  is the  $\tau_3$ -name of the node,  $e_i$ , with maximum depth among the nodes between  $z_i$  and  $z_{i+1}$  (including  $z_i$  but excluding  $z_{i+1}$ ) in preorder (we also consider the conceptual array  $E'[1..l'_1]$ , where  $E'[i] = \text{depth}(e_i)$ , but we do not store  $E'$  explicitly);
- A two-dimensional array  $M$ , where  $M[i, j]$  stores the value  $\delta$  such that  $E'[i + \delta]$  is the maximum between and including  $E'[i]$  and  $E'[i + 2^j]$ , for  $1 \leq i < l'_1$  and  $1 \leq j \leq \lceil \lg \lg n \rceil$ ;
- A table  $A_1$ , in which for each pair of nodes in each possible micro-tree, we store the  $\tau_3$ -name of the node of the maximum depth between (inclusive) this pair of nodes in preorder. This table could be used for several trees.

There are  $O(n/\lg^4 n)$  tier-1 and  $O(n/\lg n)$  tier-2 preorder changers, so  $B_1$  and  $B'_1$  can be stored in  $o(n)$  bits using Part (b) of Lemma 1.  $C_1, C'_1$  and  $E$  can also be stored in  $o(n)$  bits (each element of  $C'_1$  in  $O(\lg \lg n)$  bits). As  $M[i, j] \leq 2^{\lceil \lg \lg n \rceil}$ , we can store each  $M[i, j]$  in  $\lceil \lg \lg n \rceil$  bits, so  $M$  takes  $O(n/\lg n \times \lg \lg n \times \lg \lg n) = o(n)$  bits. As there are  $n^{1/4}$  possible micro-trees, and a micro-tree has at most  $\lceil \lg n/8 \rceil$  nodes,  $A_1$  occupies  $o(n)$  bits. Thus these auxiliary structures occupy  $o(n)$  bits.

To support the range maximum query on  $D$ , we divide the given range  $[i, j]$  into up to three subranges:  $[i, \lceil i/\lg n \rceil \lg n]$ ,  $[\lceil i/\lg n \rceil \lg n, \lfloor j/\lg n \rfloor \lg n]$  and  $[\lfloor j/\lg n \rfloor \lg n, j]$ . The result is the largest among the maximum values of these

three subranges. The range maximum query on the second subrange is supported by Sadakane’s approach (see above), so we consider only the first (the query on the third one, and the case where  $[i, j]$  is indivisible using this approach can be supported similarly).

To support range maximum query for the range  $[i, \lceil i/\lg n \rceil \lg n]$ , we first use  $B'_1$  to check whether there is a tier-2 preorder changer in this range. If not, then all the nodes in the range is in the same micro-tree. We further use  $B_1, B'_1, C_1$  and  $C'_1$  to retrieve the micro-tree by performing rank/select operations, and then use  $A_1$  to compute the result in constant time.

If there are one or more tier-2 preorder changes in  $[i, \lceil i/\lg n \rceil \lg n]$ , let node  $z_u$  be the first one and  $z_v$  be the last. We further divide this range into three subranges:  $[i, z_u], [z_u, z_v]$  and  $[z_v, \lceil i/\lg n \rceil \lg n]$ . We can compute the maximum values in the first and the third subranges using the method described in the last paragraph, as the nodes in either of them are in the same micro-tree. To perform range maximum query on  $D$  with the range  $[z_u, z_v]$ , by the definition of  $E'$ , we only need to perform the query on  $E'$  with range  $[u, v-1]$ . we observe that  $[u, v] = [u, u+2^s] \cup [v-2^s, v]$ , where  $s = \lceil \lg(v-1-u) \rceil$ . As  $v-u < z_v - z_u < \lg n$ ,  $s \leq \lceil \lg \lg n \rceil$ . Thus using  $M[u, s]$  and  $M[v-2^s, s]$ , we can retrieve from  $E$  the  $\tau_3$ -names of the nodes corresponding to the maximum values of  $E'$  in  $[u, u+2^s]$  and  $[v-2^s, v]$ , respectively. We further retrieve these two nodes using  $B_1, B'_1, C_1$  and  $C'_1$ , and the node with the larger depth is the one with the maximum depth in range  $[z_u, z_v]$ .

### 3.2 LCA and distance

We pre-compute a tier-1 macro tree as follows. First remove any node that is not a mini-tree root. For any two remaining nodes  $x$  and  $y$ , there is an edge from  $x$  to  $y$  iff among the remaining nodes,  $x$  is the nearest ancestor of  $y$  in  $T$ . Given a tree of  $n$  nodes, Bender *et al.* (2) showed how to support LCA using  $O(n \lg n)$  additional bits (3) for any tree representation. We store the tier-1 macro tree using the representation by Geary *et al.* (7) and then build the structures by Bender *et al.* (2) to support LCA and the operations listed in the column old TC of Table 1 in  $O(1)$  time. As the tier-1 macro tree has  $O(n/\lg^4 n)$  nodes, this costs  $O(n/\lg^4 n \times \lg n) = o(n)$  bits. Similarly, for each mini-tree, we pre-compute a tier-2 macro tree for the micro-tree roots, and store it using the same approach so that all the tier-2 macro trees occupy  $o(n)$  bits in total. We also construct a table  $A_2$  to store, for each possible micro-tree and each pair of nodes in it (indexed by their  $\tau_3$ -names), the  $\tau_3$ -name of their lowest common ancestor. Similarly to the analysis in Section 3.1,  $A_2$  occupies  $o(n)$  bits.

Figure 2 presents the algorithm to compute LCA. The correctness is straightforward and it clearly takes  $O(1)$  time. With the support for LCA and `depth`, the support for `distance` is trivial.

---

<sup>3</sup> They did not analyze the space cost in (2), but it is easy to verify that the cost is  $O(n \lg n)$  bits.

**Algorithm**  $\text{LCA}(x, y)$ 

1. If  $x$  and  $y$  are in the same micro-tree, retrieve their LCA using a constant-time lookup on  $A_2$  and return.
2. If  $x$  and  $y$  are not in the same micro-tree, but are in the same mini-tree, retrieve the roots,  $u$  and  $v$ , of the micro-trees that  $x$  and  $y$  are in, respectively.
3. If  $u = v$ , return  $u$  as the result.
4. If  $u \neq v$ , retrieve their lowest common ancestor,  $w$ , in the tier-2 macro tree.
5. Retrieve the two children,  $i$  and  $j$ , of  $w$  in the tier-2 macro tree that are ancestors of  $x$  and  $y$ , respectively using `depth` and `level_anc`. Then retrieve the parents,  $k$  and  $l$ , of  $i$  and  $j$  in  $T$ , respectively.
6. If  $k$  and  $l$  are in two different micro-trees, return  $w$  as the result. Otherwise, return  $\text{LCA}(k, l)$ .
7. If  $x$  and  $y$  are in two different mini-trees, retrieve the roots,  $p$  and  $q$ , of the two different mini-trees, respectively.
8. If  $p = q$ , return  $p$  as the result. Otherwise, similarly to Steps 4 and 5, retrieve two nodes  $a$  and  $b$ , such that  $a$  and  $b$  are the children of the lowest common ancestor,  $c$ , of  $p$  and  $q$  in the tier-1 macro tree, and they are also the ancestors of  $p$  and  $q$ , respectively. Retrieve the parents,  $r$  and  $s$ , of  $p$  and  $q$  in  $T$ , respectively.
9. If  $r$  and  $s$  are in two different mini-trees, return  $c$ . Otherwise, return  $\text{LCA}(r, s)$ .

**Fig. 2.** An algorithm for computing LCA**3.3 leaf\_rank, leftmost\_leaf, leaf\_size and leaf\_select**

**Definition 2.** Each leaf of a mini-tree (or micro-tree) is a *pseudo leaf* of the original tree  $T$ . A pseudo leaf that is also a leaf of  $T$  is a *real leaf*. Given a mini-tree (or micro-tree), we mark the first real leaf and each real leaf visited after an edge that leaves the mini-tree (or micro-tree). These nodes are called *tier-1* (or *tier-2*) *marked leaves*.

For example, in Figure [11](#), nodes 6, 11 and 15 are pseudo leaves of micro-tree  $\mu_1^2$ , among which nodes 6 and 15 are real leaves, while node 11 is not. Nodes 23 and 29 are tier-2 marked leaves. We observe the following property.

*Property 1.* Given a mini-tree (or micro-tree) and a pair of its tier-1 (or tier-2) marked leaves such that there is no marked leaf between them in preorder, the real leaves visited in preorder between these two leaves (including the left one but excluding the right) have the property that, when listed from left to right, their `leaf_ranks` are consecutive integers. The real leaves that are to the right of (and including) the rightmost marked leaf have the same property.

Observe that each tier-1 marked leaf (except perhaps the first real leaf in a mini-tree) corresponds to an edge that leaves its mini-tree (which in turn corresponds to a unique mini-tree root). Therefore, the number of tier-1 marked leaves is at most twice as many as the number of mini-trees, which is  $O(n/\lg^4 n)$ . We thus store for each mini-tree  $\mu^i$ , the ranks of its tier-1 marked leaves. Similarly, the number of tier-2 marked leaves is at most twice the number of micro-trees.

For each tier-2 marked leaf  $x$ , let  $y$  be the last tier-1 marked leaf visited before  $x$  during a preorder traversal (if  $x$  is a tier-1 marked leaf then we set  $y = x$ ). We compute the difference of `leaf_rank`( $x$ ) and `leaf_rank`( $y$ ) (bounded by the number of nodes in  $x$ 's mini-tree) and store for each micro-tree, the list of such values for its tier-2 marked leaves. With additional bit vectors to mark the positions of these marked leaves, and an  $o(n)$ -bit table to lookup the number of nodes between two pseudo leaves in a micro-tree, we can support `leaf_rank` in constant time by Property [11](#). We omit the details.

To support `leftmost_leaf` and `rightmost_leaf`, given a node  $x$  with preorder number  $i$ , postorder number  $j$ , and  $m$  descendants, we observe that the postorder number of its left-most leaf is  $j - m$ , and the preorder number of its rightmost leaf is  $i + m$ . Thus the support of these two operations follows the constant-time support for `node_rankPRE`, `node_rankPOST`, `node_selectPRE` and `node_selectPOST`. With the constant-time support for `leaf_rank`, `leftmost_leaf` and `rightmost_leaf`, we can also support `leaf_size` in constant time.

To support `leaf_select`, we observe that given a leaf  $x$  that is not a tier-1 (or tier-2) marked leaf, if the closest tier-1 (or tier-2) marked leaf to the left is node  $y$  (or node  $z$ ), then  $\tau_1(x) = \tau_1(y)$  (or  $\tau_2(x) = \tau_2(z)$ ). This property allows us to support `leaf_select` in  $O(1)$  time with  $o(n)$  additional bits, using an approach similar to that of Section [3.1](#). We omit the details.

### 3.4 `node_rankDFUDS` and `node_selectDFUDS`

We use the following formula proposed by Barbay *et al.* [\[1\]](#): `node_rankDFUDS`( $x$ ) =  $\begin{cases} \text{child\_rank}(x) - 1 + \text{node\_rank}_{\text{DFUDS}}(\text{child}(\text{parent}(x), 1)) & \text{if } \text{child\_rank}(x) > 1; \\ \text{node\_rank}_{\text{PRE}}(x) + \sum_{y \in \text{anc}(x) \setminus r} (\text{degree}(\text{parent}(y)) - \text{child\_rank}(y)) & \text{otherwise.} \end{cases}$  where `parent`( $x$ ) = `level_anc`( $x, 1$ ), `anc`( $x$ ) is the set of ancestors of  $x$ , and  $r$  is the root of  $T$ . This formula reduces the support of `node_rankDFUDS` to the support of computing  $S(x) = \sum_{y \in \text{anc}(x) \setminus r} \text{degree}(\text{parent}(y)) - \text{childrank}(y)$  for any given node  $x$  [\[1\]](#). We use  $u(x)$  and  $v(x)$  to denote the roots of the mini-tree and micro-tree of node  $x$ , respectively. Then we compute  $S(x)$  as the sum of the following three values as suggested by Barbay *et al.* [\[1\]](#):  $S_1(x) = S(u(x))$ ,  $S_2(x) = S(v(x)) - S(u(x))$ , and  $S_3(x) = S(x) - S(v(x))$ . It is trivial to support the computation of  $S_1(x)$  in constant time: for each mini-tree root  $i$ , we simply precompute and store  $S(i)$  using  $O(n/\lg^3 n) = o(n)$  bits. However, we cannot do the same to support the computation of  $S_2(x)$ . The approach of Barbay *et al.* [\[1\]](#) does not solve the problem, either, because Property 1 in [\[1\]](#) does not hold. To address this problem, we extend the mini-trees using the same method used to extend micro-trees. As with the Proposition 4.1 in [\[7\]](#), we have that except for the roots of mini-trees, all the other original nodes have the property that (at least a promoted copy of) each of their children is in the same extended mini-tree as themselves. With this we can support `node_rankDFUDS`. We omit the details.

To support `node_selectDFUDS`, we first define the  $\tau^*$ -name of a node and show how to convert  $\tau^*$ -names to  $\tau$ -names. Then we show how to convert `DFUDS` numbers to  $\tau^*$ -names.

**Definition 3.** Given a node  $x$  whose  $\tau$ -name is  $\tau(x) = \langle \tau_1(x), \tau_2(x), \tau_3(x) \rangle$ , its  $\tau^*$ -name is  $\tau^*(x) = \langle \tau_1(x), \tau_2(x), \tau_3^*(x) \rangle$ , if  $x$  is the  $\tau_3^*(x)^{th}$  node of its micro-tree in DFUDS order.

For example, in Figure 11, node 29 has  $\tau$ -name  $\langle 3, 1, 5 \rangle$  and  $\tau^*$ -name  $\langle 3, 1, 4 \rangle$ . To convert the  $\tau^*$ -name of a node to its  $\tau$ -name, we only need convert its  $\tau_3^*$ -name to its  $\tau_3$ -name, in constant time using a table of size  $o(n)$ .

As with the algorithm in Section 4.3.1 of [7] supporting `node_selectPRE`, the idea of computing the  $\tau^*$ -name given a DFUDS number is to store the  $\tau^*$ -names of some of the nodes, and compute the  $\tau^*$ -names of the rest using these values.

**Definition 4.** List the nodes in DFUDS order, numbered  $1, 2, \dots, n$ . The  $i^{th}$  node in DFUDS order is a **tier-1** (or **tier-2**) DFUDS **changer** if  $i = 1$ , or if the  $i^{th}$  and  $(i - 1)^{th}$  nodes in DFUDS order are in different mini-trees (or micro-trees).

For example, in Figure 11, nodes 1, 2, 16, 3, 17, 22 and 30 are tier-1 DFUDS changers, and nodes 2, 16, 3, 6, 7, 11, 4 and others are tier-2 DFUDS changers. It is obvious that all the tier-1 DFUDS changers are also tier-2 DFUDS changers. We have the following lemma (we omit its proof because of the space constraint).

**Lemma 3.** The number of tier-1 (or tier-2) DFUDS order changers is at most four times the number of mini-trees (or micro-trees).

This allows us to convert DFUDS numbers to  $\tau^*$ -names with  $o(n)$  additional bits, using an approach similar to that of Section 3.1. We omit the details.

### 3.5 level\_leftmost, level\_rightmost, level\_succ and level\_pred

We define the  $i^{th}$  level of a tree to be the set of nodes whose depths are equal to  $i$ . To support `level_leftmost` (`level_rightmost` can be supported similarly), we first show how to compute the  $\tau_1$ -name,  $u$ , of the node `level_leftmost`( $i$ ). Let  $h$  be the height of  $T$ . We construct a bit vector  $B[1..h]$ , in which  $B[j] = 1$  iff the nodes `level_leftmost`( $j - 1$ ) and `level_leftmost`( $j$ ) are in two different mini-trees, for  $1 < j \leq h$  (we set  $B[1] = 1$ ). Let  $m$  be the number of 1s in  $B$ , and we construct an array  $C[1..m]$  in which  $C[k]$  stores the  $\tau_1$ -name of the node `level_leftmost`(`select`<sub>1</sub>( $B, k$ )). As the  $\tau_1$ -name of the node `level_leftmost`( $i$ ) is the same as that of the node `level_leftmost`(`select`<sub>1</sub>( $B, \text{rank}_1(B, i)$ )), we have that  $u = C[\text{rank}_1(B, i)]$ . To analyze the space cost of  $B$  and  $C$ , we observe that if a given value,  $p$ , occurs  $q$  times in  $C$ , then the mini-tree  $\mu^p$  has at least  $q - 1$  edges that leave  $\mu^p$ . If we map the first occurrence of  $p$  to mini-tree  $\mu^p$ , and each of the rest of the occurrences of  $p$  to a unique edge of  $\mu^p$  that leaves  $\mu^p$ , then we can map each element of  $C$ , to either to a unique mini-tree, or to a unique edge that leaves a mini-tree. Thus  $m$  is at most the number of mini-trees plus the number of edges that leave a mini-tree, which is  $O(n/\lg^4 n)$ . Hence  $B$  and  $C$  occupy  $o(n)$  bits. Similarly, we can support the computation of the  $\tau_2$ -name,  $v$ , of the node `level_leftmost`( $i$ ) in constant time using  $o(n)$  additional bits. The  $\tau_3$ -name can be computed in constant time using an  $o(n)$ -bit table.

The support for `level_succ` and `level_pred` is based on the above approach and other techniques. We omit the details.

## 4 Computing a Subsequence of BP and DFUDS

We use  $\text{BP}[1..2n]$  to denote the BP sequence. Recall that each opening parenthesis in BP corresponds to the preorder number of a node, and each closing parenthesis corresponds to the postorder. Thus the number of opening parentheses corresponding to tier-1 (or tier-2) preorder changers is  $O(n/\lg^4 n)$  (or  $O(n/\lg n)$ ), and we call them *tier-1 (or tier-2) marked opening parentheses*.

We first show how to compute the subsequence of BP starting from a tier-2 marked opening parenthesis up to the next tier-2 marked opening parenthesis. We use  $j$  and  $k$  to denote the positions of these two parentheses in BP, respectively, and thus our goal is to compute  $\text{BP}[j..k-1]$ . We construct auxiliary data structures  $B_6, B'_6, C_6$  and  $C'_6$ , which are similar to the structures in Section 3.1, except that they store  $\tau$ -names of the nodes that correspond to marked opening parentheses. We also construct a table  $A_6$ , in which for each possible micro-tree and each one of its nodes, we store the subsequence of the balanced parenthesis sequence of the micro-tree, starting from the opening parenthesis corresponding to this node, to the end of this sequence, and we also store the length of such a subsequence. These auxiliary data structures occupy  $O(n \lg \lg n / \lg n)$  bits. To compute  $\text{BP}[j..k-1]$ , we first compute, in constant time, the  $\tau$ -names of the tier-2 preorder changers,  $x$  and  $y$ , whose opening parentheses are stored in  $\text{BP}[j]$  and  $\text{BP}[k]$ , respectively, using  $B_6, B'_6, C_6$  and  $C'_6$ . If  $x$  and  $y$  are in the same micro-tree, then we can perform a constant-time lookup on  $A_6$  to retrieve the result. Otherwise,  $\text{BP}[j..k-1]$  is the concatenation of the following two sequences: the subsequence of the balanced parenthesis sequence of  $x$ 's micro-tree, starting from the opening parenthesis corresponding to  $x$ , to the end of this sequence, and zero or more closing parentheses. Thus,  $\text{BP}[j..k-1]$  can either be computed in constant time using table lookup on  $A_6$  if its length is at most  $\lg n/4$ , or any  $\lg n$  subsequence of it can be computed in constant time.

To compute any  $O(\lg n)$ -bit subsequence of BP, we conceptually divide BP into blocks of size  $\lg n$ . As any  $O(\lg n)$ -bit subsequence spans a constant number of blocks, it suffices to support the computation of a block. For a given block with  $u$  tier-2 marked opening parentheses, we can run the algorithm described in the last paragraph at most  $u+1$  times to retrieve the result. To facilitate this, we choose a function  $f(n)$  where  $f(n) = O(\lg n)$  and  $f(n) = \Omega(1)$ . We explicitly store the blocks that have  $2f(n)$  or more tier-2 marked opening parentheses, which takes at most  $2n/(\lg n \times 2f(n)) \times \lg n = n/f(n)$  bits. Thus, a block explicitly stored can be computed in  $O(1)$  time, and a block that is not can be computed in  $O(f(n))$  time as it has at most  $2f(n)$  tier-2 marked opening parentheses.

To support the computation of a word of the DFUDS sequence, recall that the DFUDS sequence can be considered as the concatenation of the DFUDS subsequences of all the nodes in preorder (See Section 1.1). Thus the techniques used above can be modified to support the computation of a subsequence of DFUDS. The main difference is that we need the notion of extended micro-trees, as they have the property that the children of each non-root original node of an extended micro-tree are all in the same extended micro-tree.



## 5 Open Problems

The first open problem is whether we can compute any  $O(\lg n)$ -bit subsequence of BP or DFUDS in constant time using  $o(n)$  additional bits for TC. Our result in Theorem 2 is in the form of time/space tradeoff and we do not know whether it is optimal. Other interesting open problems include the support of the operations that are not previously supported by BP, DFUDS or TC. One is to support rank/select operations on the level-order traversal of the tree. Another one is to support `level_leftmost` (`level_rightmost`) on an arbitrary subtree of  $T$ .

## References

- [1] Barbay, J., Rao, S.: Succinct encoding for XPath location steps. Technical Report CS-2006-10, University of Waterloo, Ontario, Canada (2006)
- [2] Bender, M.A., Farach-Colton, M.: The LCA problem revisited. In: Proc. 7th Latin American Theoretical Informatics Symp., pp. 88–94 (2000)
- [3] Benoit, D., Demaine, E.D., Munro, J.I., Raman, R., Raman, V., Rao, S.S.: Representing trees of higher degree. *Algorithmica* 43(4), 275–292 (2005)
- [4] Benoit, D., Demaine, E.D., Munro, J.I., Raman, V.: Representing trees of higher degree. In: Dehne, F., Gupta, A., Sack, J.-R., Tamassia, R. (eds.) WADS 1999. LNCS, vol. 1663, pp. 169–180. Springer, Heidelberg (1999)
- [5] Chiang, Y.-T., Lin, C.-C., Lu, H.-I.: Orderly spanning trees with applications to graph encoding and graph drawing. In: Proc. 12th ACM-SIAM Symp. Discrete Algorithms, pp. 506–515 (2001)
- [6] Clark, D.R., Munro, J.I.: Efficient suffix trees on secondary storage. In: Proc. 7th ACM-SIAM Symp. Discrete Algorithms, pp. 383–391 (1996)
- [7] Geary, R.F., Raman, R., Raman, V.: Succinct ordinal trees with level-ancestor queries. *ACM Trans. Algorithms* 2(4), 510–534 (2006)
- [8] Jacobson, G.: Space-efficient static trees and graphs. In: Proc. 30th IEEE Symp. Found. Comput. Sci., pp. 549–554 (1989)
- [9] Jansson, J., Sadakane, K., Sung, W.-K.: Ultra-succinct representation of ordered trees. In: Proc. 18th ACM-SIAM Symp. Discrete Algorithms, pp. 575–584 (2007)
- [10] Lu, H.-I., Yeh, C.-C.: Balanced parentheses strike back. Accepted to *ACM Trans. Algorithms* (2007)
- [11] Munro, J.I., Raman, V.: Succinct representation of balanced parentheses and static trees. *SIAM J. Comput.* 31(3), 762–776 (2001)
- [12] Munro, J.I., Raman, V., Rao, S.S.: Space efficient suffix trees. *J. Algorithms* 39(2), 205–222 (2001)
- [13] Munro, J.I., Rao, S.S.: Succinct representations of functions. In: Proc. 31st Int. Colloquium Automata, Languages and Programming, pp. 1006–1015 (2004)
- [14] Raman, R., Raman, V., Rao, S.S.: Succinct indexable dictionaries with applications to encoding  $k$ -ary trees and multisets. In: Proc. 13th ACM-SIAM Symp. Discrete Algorithms, pp. 233–242 (2002)
- [15] Sadakane, K.: Succinct representations of lcp information and improvements in the compressed suffix arrays. In: Proc. 13th ACM-SIAM Symp. Discrete Algorithms, pp. 225–232 (2002)

# A Framework for Dynamizing Succinct Data Structures<sup>★</sup>

Ankur Gupta<sup>1</sup>, Wing-Kai Hon<sup>2</sup>, Rahul Shah<sup>1</sup>, and Jeffrey Scott Vitter<sup>1</sup>

<sup>1</sup> Department of Computer Sciences, Purdue University,  
West Lafayette, IN 47907–2107, USA  
{agupta, rahul, jsv}@cs.purdue.edu

<sup>2</sup> Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan  
wkhon@cs.nthu.edu.tw

**Abstract.** We present a framework to dynamize succinct data structures, to encourage their use over non-succinct versions in a wide variety of important application areas. Our framework can dynamize most state-of-the-art succinct data structures for dictionaries, ordinal trees, labeled trees, and text collections. Of particular note is its direct application to XML indexing structures that answer *subpath* queries [2]. Our framework focuses on achieving information-theoretically optimal space along with near-optimal update/query bounds.

As the main part of our work, we consider the following problem central to text indexing: Given a text  $T$  over an alphabet  $\Sigma$ , construct a compressed data structure answering the queries  $char(i)$ ,  $rank_s(i)$ , and  $select_s(i)$  for a symbol  $s \in \Sigma$ . Many data structures consider these queries for static text  $T$  [5, 3, 16, 4]. We build on these results and give the best known query bounds for the dynamic version of this problem, supporting arbitrary insertions and deletions of symbols in  $T$ .

Specifically, with an amortized update time of  $O(n^\epsilon)$ , any static succinct data structure  $D$  for  $T$ , taking  $t(n)$  time for queries, can be converted by our framework into a dynamic succinct data structure that supports  $rank_s(i)$ ,  $select_s(i)$ , and  $char(i)$  queries in  $O(t(n) + \log \log n)$  time, for any constant  $\epsilon > 0$ . When  $|\Sigma| = \text{polylog}(n)$ , we achieve  $O(1)$  query times. Our update/query bounds are near-optimal with respect to the lower bounds from [13].

## 1 Introduction

The new trend in indexing data structures is to compress and index data in one shot. The ultimate goal of these compressed indexes is to retain near-optimal query times (as if not compressed), yet still take near-optimal space (as if not an index). A few pioneer results are [6, 5, 3, 15, 4, 2]; there are many others. For compressed text indexing, see Navarro and Mäkinen’s excellent survey [11].

Progress in compressed indexing has also expanded to more combinatorial structures, such as trees and subsets. For these *succinct* data structures, the

---

<sup>★</sup> Support was provided in part by the National Science Foundation through research grants CCF–0621457 and IIS–0415097.

emphasis is to store them in terms of the information-theoretic (combinatorial) minimum required space with fast query times [15,9,7]. Compressed text indexing makes heavy use of succinct data structures for set data, or *dictionaries*.

The vast majority of succinct data structuring work is concerned largely with static data. Although the space savings is large, the main deterrent to a more ubiquitous use of succinct data structures is their notable lack of support for dynamic operations. Many settings require indexing and query functionality on dynamic data: XML documents, web pages, CVS projects, electronic document archives, etc. For this type of data, it can be prohibitively expensive to rebuild a static index from scratch each time an update occurs. The goal is then to answer queries efficiently, perform updates in a reasonable amount of time, and *still* maintain a compressed version of the dynamically-changing data.

In that vein, there have been some results on dynamic succinct bitvectors (dictionaries) [14,8,12]. However, these data structures either perform queries in far from optimal time (in query-intensive environments), or allow only a limited range of dynamic operations (“flip” operations only). Here, we consider the more general update operations consisting of arbitrary insertion and deletion of bits, which is a central challenge in dynamizing succinct data structures for a variety of applications. We define the *dynamic text dictionary* problem: Given a dynamic text  $T$  of  $n$  symbols drawn from an alphabet  $\Sigma$ , construct a data structure (index) that allows the following operations for any symbol  $s \in \Sigma$ :

- $rank_s(i)$  tells the number of  $s$  symbols up to the  $i$ th position in  $T$ ;
- $select_s(i)$  gives the position in  $T$  of the  $i$ th  $s$ ;
- $char(i)$  returns the symbol in the  $i$ th position of  $T$ ;
- $insert_s(i)$  inserts  $s$  before the position  $i$  in  $T$ ;
- $delete(i)$  deletes the  $i$ th symbol from  $T$ .

When  $|\Sigma| = 2$ , the above problem is called the *dynamic bit dictionary* problem. For the static case, [15] solves the bit dictionary problem using  $nH_0 + o(n)$  bits of space and answers rank and select queries in  $O(1)$  time, where  $H_0$  is the 0th order empirical entropy of the text  $T$ . The best known time bounds for the dynamic problem are given by [12], achieving  $O(\log n)$  for all operations.  $\square$

The text dictionary problem is a key tool in text indexing data structures. For the static case, Grossi et al. [5] present a wavelet tree structure that answers queries in  $O(\log |\Sigma|)$  time and takes  $nH_0 + o(n \log |\Sigma|)$  bits of space. Golynski et al. [4] improve the query bounds to  $O(\log \log |\Sigma|)$  time, although they take more space, namely,  $n \log |\Sigma| + o(n \log |\Sigma|)$  bits of space. Nevertheless, their data structure presents the best query bounds for this problem.

Developing a dynamic text dictionary based on the wavelet structure can be done readily using dynamic bit dictionaries (as is done in [12]) since updates to a particular symbol  $s$  only affect the data structures for  $O(\log |\Sigma|)$  groups of symbols according to the hierarchical decomposition of the alphabet  $\Sigma$ . The solution to this problem is given by Mäkinen and Navarro [12], with an update/query bound of  $O(\log n \log |\Sigma|)$ . These bounds are far from optimal, especially in query-intensive settings. On the other hand, the best known query bounds for static

<sup>1</sup> There is another data structure proposed in [8], requiring non-succinct space.

text dictionaries are given by [4], which treats each symbol in  $\Sigma$  individually; an update to symbol  $s$  could potentially affect  $\Sigma$  different data structures, and thus may be hard to dynamize.

We list the following contributions of our paper:

- We develop a general framework to dynamize many succinct data structures like ordinal trees, labeled trees, dictionaries, and text collections. Our framework can transform any static succinct data structure  $D$  for a text  $T$  into a dynamic succinct data structure. Precisely, if  $D$  supports  $rank_s$ ,  $select_s$ , and  $char$  queries in  $O(t(n))$  time and takes  $s(n)$  bits of space, the dynamic data structure supports queries in  $O(t(n) + \log \log n)$  time and updates in amortized  $O(n^\epsilon)$  time and takes just  $s(n) + o(n)$  bits of space.
- Our results represent near-optimal tradeoffs for update/query times for the dynamic text (and bit) dictionary problem. (For lower bound, see [13].)
- We provide the first succinct data structure for the *dynamic bit dictionary* problem. Our data structure takes  $nH_0 + o(n)$  bits of space and requires  $O(\log \log n)$  time to support  $rank_s$ ,  $select_s$ , and  $char$  queries while supporting updates to the text  $T$  in amortized  $O(n^\epsilon)$  time.
- We provide the first near-optimal result for the *dynamic text dictionary* problem on a dynamic text  $T$ . Our data structure requires  $n \log |\Sigma| + o(n \log |\Sigma|)$  bits of space and supports queries in  $O(\log \log n)$  time and updates in  $O(n^\epsilon)$  time. When  $|\Sigma| = \text{polylog}(n)$ , we can improve our query time to  $O(1)$ .
- Our framework can dynamize succinct data structures for labeled trees, text collections, and XML documents.

## 2 Preliminaries

We summarize several important static structures that we will use in achieving the dynamic results. The proofs of their construction are omitted due to space constraints. In the rest of this paper, we refer to a static bit or text dictionary  $D$ , that requires  $s(n)$  bits and answers queries in  $t(n)$  time.

**Lemma 1 ([15]).** *For a bitvector (i.e.,  $|\Sigma| = 2$ ) of length  $n$ , there exists a static data structure  $D$  called RRR solving the bit dictionary problem supporting  $rank$ ,  $select$ , and  $char$  queries in  $t(n) = O(1)$  time using  $s(n) = nH_0 + O(n \log \log n / \log n)$  bits of space, while taking  $O(n)$  time to construct.  $\square$*

**Lemma 2 ([5]).** *For a text  $T$  of length  $n$  drawn from alphabet  $\Sigma$ , there exists a static data structure  $D$  called the wavelet tree solving the text dictionary problem supporting  $rank_s$ ,  $select_s$ , and  $char$  queries in  $t(n) = O(\log |\Sigma|)$  time using  $s(n) = nH_0 + o(n \log |\Sigma|)$  bits of space, while taking  $O(nH_0)$  time to construct. When  $|\Sigma| = \text{polylog}(n)$ , we can support queries in  $t(n) = O(1)$  time.  $\square$*

**Lemma 3 ([4]).** *For a text  $T$  of length  $n$  drawn from alphabet  $\Sigma$ , there exists a static data structure  $D$  called GMR that solves the text dictionary problem supporting  $select_s$  queries in  $t_1(n) = O(1)$  time and  $rank$  and  $char$  queries in  $t_2(n) = O(\log \log |\Sigma|)$  time using  $s(n) = n \log |\Sigma| + o(n \log |\Sigma|)$  bits of space, while taking  $O(n \log n)$  time to construct.  $\square$*

We also use the following static data structure called prefix-sum (PS) as a building block for achieving our dynamic result. Suppose we are given a non-negative integer array  $A[1..t]$  such that  $\sum_i A[s_i] \leq n$ . We define the partial sums  $P[i] = \sum_{j=1}^i A[j]$ . Note that  $P$  is a sorted array, such that  $0 \leq P[i] \leq P[j] \leq n$  for all  $i < j$ . A prefix-sum (PS) structure on  $A$  is a data structure that supports the following operations:

- $sum(j)$  returns the partial sum  $P[j]$ ;
- $findsum(i)$  returns the index  $j$  such that  $sum(j) \leq i < sum(j+1)$ .

**Lemma 4.** *Let  $A[1 \dots t]$  be a non-negative integer array such that  $\sum_i A[i] \leq n$ . There exists a data structure PS on  $A$  that supports  $sum$  and  $findsum$  in  $O(\log \log n)$  time using  $O(t \log n)$  bits of space and can be constructed in  $O(t)$  time. In the particular case where  $x \leq A[i] \leq cx$  for all  $i$ , where  $x$  is a positive integer and  $c \geq 1$  is a positive constant integer,  $sum$  and  $findsum$  can be answered in  $O(1)$  time.  $\square$*

We also make use of a data structure called the Weight Balanced B-tree (WBB tree), which was used in [14,8]. We use this structure with Lemma 4 to achieve  $O(1)$  time. A WBB tree is a B-tree defined with a *weight-balance condition*. A weight-balance condition means that for any node  $v$  at level  $i$ , the number of leaves in  $v$ 's subtree is between  $0.5b^i + 1$  and  $2b^i - 1$ , where  $b$  is the fanout factor. Insertions and deletions on the WBB tree can be performed in amortized  $O(\log_b n)$  time while maintaining the weight-balance condition.

We use the WBB tree since it ensures that  $x \leq A[i] \leq cx$  where  $c$  is a positive constant integer, thus allowing constant-time search at each node. However, a simple B-tree would require  $O(\log \log n)$  time in this situation. Also, WBB trees are a crucial component of the onlyX structure, described in Section 3.3. WBB trees are also used in Section 3.1 (although B-trees could be used here).

### 3 Data Structures

Our solution is built with three main data structures:

- *BitIndel*: bitvector supporting insertion and deletion, described in Section 3.1;
- *StaticRankSelect*: static text dictionary structure supporting  $rank_s$ ,  $select_s$ , and  $char$  on a text  $T$ ;
- *onlyX*: non-succinct dynamic text dictionary, described in Section 3.3.

We use StaticRankSelect to maintain the original text  $T$ ; we can use any existing structure such as the wavelet tree or GMR mentioned in Section 2. For ease of exposition, unless otherwise stated, we shall use GMR [4] in this section. We keep track of newly inserted symbols  $N$  in onlyX such that after every  $O(n^{1-\epsilon} \log n)$  update operations performed, updates are merged with the StaticRankSelect structure. Thus, onlyX never contains more than  $O(n^{1-\epsilon} \log n)$  symbols. We maintain onlyX using  $O(n^{1-\epsilon} \log^2 n) = o(n)$  bits of space. Finally, since merging  $N$  with  $T$  requires  $O(n \log n)$  time, we arrive at an amortized  $O(n^\epsilon)$  time for updating these data structures. BitIndel is used to translate positions  $p_t$  from the old text  $T$  to the new positions  $p_t$  from the current text  $\hat{T}$ . (We maintain  $\hat{T}$  implicitly through the use of BitIndel structures, StaticRankSelect, and onlyX.)

### 3.1 Bitvector Dictionary with Indels: BitIndel

In this section, we describe a data structure (BitIndel) for a bitvector  $B$  of original length  $n$  that can handle insertions and deletions of bits anywhere in  $B$  while still supporting *rank* and *select* on the updated bitvector  $B'$  of length  $n'$ . The space of the data structure is  $n'H_0 + o(n')$ . When  $n' = O(n)$ , our structure supports these updates in  $O(n^\epsilon)$  time and *rank* and *select* queries in  $O(\log \log n)$  time. (In [8], Hon et al. propose a non-succinct BitIndel structure taking  $n' + o(n')$  bits of space.)

Formally, we define the following update operations that we support on the current bitvector  $B'$  of length  $n'$ : *insert<sub>b</sub>(i)* inserts the bit  $b$  in the  $i$ th position, *delete(i)* deletes the bit located in the  $i$ th position, and *flip(i)* flips the bit in the  $i$ th position.

We defer the details until the full paper. The idea is to use a B-tree over  $\Theta(n^\epsilon)$ -sized chunks of the bitvector, which are stored using an RRR structure. This B-tree is constant-height and needs prefix-sum data structures in its internal nodes for fast access.

**Lemma 5.** *Given a bitvector  $B'$  with length  $n'$  and original length  $n$ , we can create a data structure that takes  $n'H_0 + o(n')$  bits and supports *rank* and *select* in  $O((\log_n n') \log \log n)$  time, and *indel* in  $O(n^\epsilon \log_n n')$  time. When  $n' = O(n)$ , our time bounds become  $O(\log \log n)$  and  $O(n^\epsilon)$  respectively.  $\square$*

The prefix sum data structure used inside the B-tree is the main bottleneck to query times, allowing us only  $O(\log \log n)$  time access. However, if we store three WBB-trees, then separately in each of them the special condition from Lemma 4 can be met allowing us  $O(1)$  queries on prefix sum structures. This allows us to obtain the following lemma.

**Lemma 6.** *Given a bitvector  $B'$  with length  $n'$  and original length  $n$ , we can create a data structure that takes  $3n'H_0 + o(n')$  bits and supports *rank* and *select* in  $O(\log_n n')$  time, and *indel* in  $O(n^\epsilon \log_n n')$  amortized time. When  $n' = O(n)$ , our time bounds become  $O(1)$  and  $O(n^\epsilon)$  respectively.  $\square$*

If we change our BitIndel structure such that the bottom-level RRR [15] data structures are built on  $[\log^2 n, 2 \log^2 n]$  bits each and set the B-tree fanout factor  $b = 2$ , we can obtain  $O(\log n)$  update time with  $O(\log n)$  query time. In this sense, our BitIndel data structure is a generalization of [12].

### 3.2 Insert-X-Delete-Any: Inx

Let  $x$  be a symbol other than those in alphabet  $\Sigma$ . In this section, we describe a data structure on a text  $T$  of length  $n$  supporting *rank<sub>s</sub>* and *select<sub>s</sub>* that can handle *delete(i)* and *insert<sub>x</sub>(i)*. That is, only  $x$  can be inserted to  $T$ , while any characters can be deleted from  $T$ . Notice that insertions and deletions will affect the answers returned for symbols in the alphabet  $\Sigma$ . For example,  $T$  may be  $abcaab$ , where  $\Sigma = \{a, b, c\}$ . Here,  $rank_a(4) = 2$  and  $select_a(3) = 5$ . Let  $\hat{T}$  be the current text after some number of insertions and deletions of symbol  $x$ .

Initially,  $\hat{T} = T$ . After some insertions, the current  $\hat{T}$  may be `axxxbcaxabx`. Notice that  $rank_a(4) = 1$  and  $select_a(3) = 9$ . We represent  $\hat{T}$  by the text  $T'$ , such that when the symbols of the original text  $T$  are deleted, each deleted symbol is replaced by a special symbol `d` (whereas if  $x$  is deleted, it is just deleted from  $T'$ ). Continuing the example, after some deletions of symbols from  $T$ ,  $T'$  may be `axxxddaxabx`. Notice that  $rank_a(4) = 1$  and  $select_a(3) = 7$ .

We define an *insert vector*  $I$  such that  $I[i] = 1$  if and only if  $T'[i] = x$ . Similarly, we define a *delete vector*  $D$  such that  $D[i] = 1$  if and only if  $T'[i] = d$ . We also define a delete vector  $D_s$  for each symbol  $s$  such that  $D_s[i] = 1$  if and only if the  $i$ th  $s$  in the original text  $T$  was deleted. The text  $T'$  is merely a conceptual text: we refer to it for ease of exposition but we actually maintain  $\hat{T}$  instead.

To store  $\hat{T}$ , we store  $T$  using the StaticRankSelect data structure and store all of the  $I$ ,  $D$ ,  $D_s$  bitvectors using the constant time BitIndel structure. Now, we describe  $\hat{T}.insert_x(i)$ ,  $\hat{T}.delete(i)$ ,  $\hat{T}.rank_s(i)$ , and  $\hat{T}.select_s(i)$ :

**$\hat{T}.insert_x(i)$ .** First, we convert position  $i$  in  $\hat{T}$  to its corresponding position  $i'$  in  $T'$  by computing  $i' = D.select_0(i)$ . Then we must update our various vectors. We perform  $I.insert_1(i')$  on our insert vector, and  $D.insert_0(i')$  on our delete vector.

**$\hat{T}.delete(i)$ .** First, we convert position  $i$  in  $\hat{T}$  to its corresponding position  $i'$  in  $T'$  by computing  $i' = D.select_0(i)$ . If  $i'$  is newly-inserted (i.e.,  $I[i'] = 1$ ), then we perform  $I.delete(i')$  and  $D.delete(i')$  to reverse the insertion process from above. Otherwise, we first convert position  $i'$  in  $T'$  to its corresponding position  $i''$  in  $T$  by computing  $i'' = I.rank_0(i')$ . Let  $s = T.char(i'')$ . Finally, to delete the symbol, we perform  $D.flip(i')$  and  $D_s.flip(j)$ , where  $j = T.rank_s(i'')$ .

**$\hat{T}.rank_s(i)$ .** First, we convert position  $i$  in  $\hat{T}$  to its corresponding position  $i'$  in  $T'$  by computing  $i' = D.select_0(i)$ . If  $s = x$ , return  $I.rank_1(i')$ . Otherwise, we first convert position  $i'$  in  $T'$  to its corresponding position  $i''$  in  $T$  by computing  $i'' = I.rank_0(i')$ . Finally, we return  $D_s.rank_0(j)$ , where  $j = T.rank_s(i'')$ .

**$\hat{T}.select_s(i)$ .** If  $s = x$ , compute  $j = I.select_1(i)$  and return  $D.rank_0(j)$ . Otherwise, we compute  $k = D_s.select_0(i)$  to determine  $i$ 's position among the  $s$  symbols from  $T$ . We then compute  $k' = T.select_s(k)$  to determine its original position in  $T$ . Now the position  $k'$  from  $T$  needs to be mapped to its appropriate location in  $\hat{T}$ . Similar to the first case, we perform  $k'' = I.select_0(k')$  and return  $D.rank_0(k'')$ , which corresponds to the right position of  $\hat{T}$ .

**$\hat{T}.char(i)$ .** First, we convert position  $i$  in  $\hat{T}$  to its corresponding position  $i'$  in  $T'$  by computing  $i' = D.select_0(i)$ . If  $I[i'] = 1$ , return `x`. Otherwise, we convert position  $i'$  in  $T'$  to its corresponding position  $i''$  in  $T$  by computing  $i'' = I.rank_0(i')$  and return  $T.char(i'')$ .

**Space and Time.** As can be seen, each of the *rank* and *select* operations requires a constant number of accesses to BitIndel and StaticRankSelect structures, thus taking  $O(1)$  time to perform. The *indel* operations require  $O(n^\epsilon)$  update time, owing to the BitIndel data structure. The space required for the

above data structures comes from the StaticRankSelect structure, which requires  $s(n) = O(n \log |\Sigma| + o(n \log |\Sigma|))$  bits of space, and the many BitIndel structures, whose space can be bounded by  $3 \log \binom{n'}{n} + 6 \log \binom{n'}{n''} + o(n') + O((n'/n^\epsilon) \log n')$  bits where  $n''$  is number of deletes. If  $n''$  and  $n' - n$  are bounded by  $n^{1-\epsilon}$ , then this expression is  $o(n)$  bits.

**Theorem 1.** *Let  $T$  be a dynamic text of original length  $n$  and current length  $n'$ , with characters drawn from an alphabet  $\Sigma$ . Let  $n''$  be the number of deletions. If the number of updates is  $O(n^{1-\epsilon})$ , We can create a data structure using GMR that takes  $n \log |\Sigma| + o(n \log |\Sigma|)$  bits of space and supports  $rank_s(i)$  and  $select_s(i)$  in  $O(1)$  time and  $insert_x(i)$  and  $delete_s(i)$  in  $O(n^\epsilon)$  time.  $\square$*

### 3.3 onlyX-structure

Let  $T$  be the dynamic text that we want to maintain, where symbols of  $T$  are drawn from alphabet  $\Sigma$ . Let  $n'$  be the current length of  $T$ , and we assume that  $n' = O(n)$ . In this section, we describe a data structure for maintaining a dynamic array of symbols that supports  $rank_s$  and  $select_s$  queries in  $O((\log_n n')(t(n) + \log \log n))$  time, for any fixed  $\epsilon$  with  $0 < \epsilon < 1$ ; here, we assume that the maximum number of symbols in the array is  $O(n)$ . Our data structure takes  $O(n' \log n)$  bits; for each update (i.e., insertion or deletion of a symbol), it can be done in amortized  $O(n^\epsilon)$  time.

We describe how to apply the WBB tree to maintain  $T$  while supporting  $rank_s$  and  $select_s$  efficiently, for any  $s \in \Sigma$ <sup>2</sup>. In particular, we choose  $\epsilon < 1$  and store the symbols of  $T$  in a WBB  $W$  with fanout factor  $b = n^\delta$  where  $\delta = \epsilon/2$  such that the  $i$ th (leftmost) leaf of  $W$  stores  $T[i]$ . Each node at level 1 will correspond to a substring of  $T$  with  $O(b)$  symbols, and we will maintain a static text dictionary for that substring so that  $rank_s$  and  $select_s$  are computed for that substring in  $t(n) = O(\log \log |\Sigma|)$  time. In each level- $\ell$  node  $v_\ell$  with  $\ell \geq 2$ , we store an array  $size$  such that  $size[i]$  stores the number of symbols in the subtree of its  $i$ th (leftmost) child. To have fast access to this information at each node, we build a PS structure to store  $size$ . Also, for each symbol  $s$  that appears in the subtree of  $v_\ell$ ,  $v_\ell$  is associated with an  $s$ -structure, which consists of three arrays:  $pos_s$ ,  $num_s$ , and  $ptr_s$ . The entry  $pos_s[i]$  stores the index of  $v_\ell$ 's  $i$ th leftmost child whose subtree contains  $s$ . The entry  $num_s[i]$  stores the number of  $s$  in  $v_\ell$ 's  $i$ th leftmost child whose subtree contains  $s$ . The entry  $ptr_s[i]$  stores a pointer to the  $s$ -structure of  $v_\ell$ 's  $i$ th leftmost child whose subtree contains  $s$ .

The arrays in each  $s$ -structure ( $size_s$ ,  $pos_s$ , and  $num_s$ ) are stored using a PS data structure so that we can support  $O(\log \log n)$ -time  $sum$  and  $findsum$  queries in  $size_s$  or  $num_s$ , and  $O(\log \log n)$ -time  $rank$  and  $select$  queries in  $pos_s$ . (These  $rank$  and  $select$  operations are analogous to  $sum$  and  $findsum$  queries, but we

<sup>2</sup> One may think of using a B-tree instead of a WBB-tree. However, in our design, a particular node in the WBB tree will need to store auxiliary information about every symbol in the subtree under that node. In the worst case, this auxiliary information will be as big as the size of the subtree. If we use a B-tree, the cost of updating a particular node cannot be bounded by  $O(n^\epsilon)$  time in the amortized case.



refer to them as *rank* and *select* for ease of exposition.) The list  $ptr_s$  is stored in a simple array.

We also maintain another B-tree  $B$  with fanout  $n^\delta$  such that each leaf  $\ell_s$  corresponds to a symbol  $s$  that is currently present in the text  $T$ . Each leaf stores the number of (nonzero) occurrences of  $s$  in  $T$ , along with a pointer to its corresponding  $s$ -structure in the root of  $W$ . The height of  $B$  is  $O(\log_{n^\epsilon} |\Sigma|) = O(1)$ , since we assume  $|\Sigma| \leq n$ .

**Answering  $char(i)$ .** We can answer this query in  $O(\log \log n)$  time by maintaining a B-tree with fanout  $b = n^\delta$  over the text. We call this tree the *text B-tree*.

**Answering  $rank_s(p)$ .** Recall that  $rank_s(p)$  tells the number of occurrences of  $s$  in  $T[1..p]$ . We first query  $B$  to determine if  $s$  occurs in  $T$ . If not, return 0. Otherwise, we follow the pointer from  $B$  to its  $s$ -structure. We then perform  $r.size_s.findsum(p)$  to determine the child  $c_i$  of root  $r$  from  $W$  that contains  $T[p]$ . Suppose that  $T[p]$  is in the subtree rooted at the  $i$ th child  $c_i$  of  $r$ . Then,  $rank_s$  consists of two parts: the number of occurrences  $m_1 = r.num_s.sum(j)$  (with  $j = r.pos_s.rank(i - 1)$ ) in the first  $i - 1$  children of  $r$ , and  $m_2$ , the number of occurrences of  $s$  in  $c_i$ . If  $r.pos_s.rank(i) \neq j + 1$  ( $c_i$  contains no  $s$  symbols), return  $m_1$ . Otherwise, we retrieve the  $s$ -structure of  $c_i$  by its pointer  $r.ptr[j + 1]$  and continue counting the remaining occurrences of  $s$  before  $T[p]$  in the WBB tree  $W$ . We will eventually return  $m_1 + m_2$ .

The above process either (i) stops at some ancestor of the leaf of  $T[p]$  whose subtree does not contain  $s$ , in which case we can report the desired rank, or (ii) it stops at the level-1 node containing  $T[p]$ , in which case the number of remaining occurrences can be determined by a  $rank_s$  query in the static text dictionary in  $t(n) = O(\log \log |\Sigma|)$  time. Since it takes  $O(\log \log n)$  time to check the B-tree  $B$  at the beginning, and it takes  $O(\log \log n)$  time to descend each of the  $O(1)$  levels in the WBB-tree to count the remaining occurrences, the total time is  $O(\log \log n)$ .

**Answering  $select_s(j)$ .** Recall that  $select_s(j)$  tells the number of symbols (inclusive) before the  $j$ th occurrence of  $s$  in  $T$ . We follow a similar procedure to the above procedure for  $rank_s$ . We first query  $B$  to determine if  $s$  occurs at least  $j$  times in  $T$ . If not, we return  $-1$ . Otherwise, we discover the  $i$ th child  $c_i$  of root  $r$  from  $W$  that contains the  $j$ th  $s$  symbol. We compute  $i = r.pos_s.select(r.num_s.findsum(j))$  to find out  $c_i$ .

Then,  $select_s$  consists of two parts: the number of symbols  $m_1 = r.size.sum(i)$  in the first  $i - 1$  children of  $r$ , and  $m_2$ , the number of symbols in  $c_i$  before the  $j$ th  $s$ . We retrieve the  $s$ -structure of  $c_i$  by its pointer  $r.ptr[r.num_s.findsum(j)]$  and continue counting the remaining symbols on or before the  $j$ th occurrence of  $s$  in  $T$ . We will eventually return  $m_1 + m_2$ . The above process will stop at the level-1 node containing the  $j$ th occurrence of  $s$  in  $T$ , in which case the number of symbols on or before it maintained by this level-1 node can be determined by a  $select_s$  query in the static text dictionary in  $t(n) = O(\log \log |\Sigma|)$  time. With similar time analysis as in  $rank_s$ , the total time is  $O(\log \log n)$ .

**Updates.** We can update the text B-tree in  $O(n^\epsilon)$  time. We use a naive approach to handle updates due to the insertion or deletion of symbols in  $T$ : For each list in the WBB-tree and for each static text dictionary that is affected, we rebuild it from scratch. In the case that no split, merge, or merge-then-split operation occurs in the WBB-tree, an insertion or deletion of  $s$  at  $T[p]$  will affect the static text dictionary containing  $T[p]$ , and two structures in each ancestor node of the leaf containing  $T[p]$ : the *size* array and the  $s$ -structure corresponding to the inserted (deleted) symbol. The update cost is  $O(n^\delta \log n) = O(n^\epsilon)$  for the static text dictionary and for each ancestor, so in total it takes  $O(n^\epsilon)$  time.

If a split, merge, or merge-then-split operation occurs at some level- $\ell$  node  $v_\ell$  in the WBB-tree, we need to rebuild the *size* array and  $s$ -structures for all newly created nodes, along with updating the *size* array and  $s$ -structures of the parent of  $v_\ell$ . In the worst case, it requires  $O(n^{(\ell+1)\epsilon} \log n)$  time. By the property of WBB trees, the amortized update takes  $O(n^\epsilon)$  time.

In summary, each update due to an insertion or deletion of symbols in  $T$  can be done in amortized  $O(n^\epsilon)$  time.

**Space complexity.** The space for the text B-tree is  $O(n \log |\Sigma| + n^{1-\epsilon} \log n)$  bits. The total space of all  $O(n^{1-\epsilon})$  static text dictionaries can be bounded by  $s(n) = O(n \log |\Sigma|)$  bits.

For the space of the  $s$ -structures, it seems like it is  $O(|\Sigma| n^{1-\epsilon} \log n)$  bits at the first glance, since there are  $O(n^{1-\epsilon})$  nodes in  $W$ . This space however is not desirable, since  $|\Sigma|$  can be as large as  $n$ . In fact, a closer look of our design reveals that each node in  $W$  only maintains  $s$ -structures for those  $s$  that appears in its subtree. In total, each character of  $T$  contributes to at most  $O(1)$   $s$ -structures, thus incurring only  $O(\log n)$  bits. The total space for  $s$  structures is thus bounded by  $O(n \log n)$  bits.

The space for the B-tree  $B$  (maintaining distinct symbols in  $T$ ) is  $O(|\Sigma| \log n)$  bits, which is at most  $O(n \log n)$  bits. In summary, the total space of the above dynamic rank-select structure is  $O(n \log n)$  bits.

Summarizing the above discussions, we arrive at the following theorem.

**Theorem 2.** *For a dynamic text  $T$  of length at most  $O(n)$ , we can maintain a data structure on  $T$  using GMR to support  $\text{rank}_s$ ,  $\text{select}_s$ , and  $\text{char}$  in  $O(t(n) + \log \log n) = O(\log \log n)$  time, and insertion/deletion of a symbol in amortized  $O(n^\epsilon)$  time. The space of the data structure is  $O(n \log n)$  bits.  $\square$*

**Theorem 3.** *Suppose that  $|\Sigma| = \text{polylog}(n)$ . For a dynamic text  $T$  of length at most  $O(n)$ , we can maintain a data structure on  $T$  using the wavelet tree to support  $\text{rank}_s$ ,  $\text{select}_s$ , and  $\text{char}$  in  $O(t(n)) = O(1)$  time, and insertion/deletion of a symbol in amortized  $O(n^\epsilon)$  time. The space of the data structure is  $O(|\Sigma| n \log n)$  bits, and the working space to perform the updates at any time is  $O(n^\epsilon)$  bits.  $\square$*

### 3.4 The Final Data Structure

Here we describe our final structure, which supports insertions and deletions of any symbol. To do this, we maintain two structures: our inX structure on  $\hat{T}$  and the

onlyX structure, where all of the new symbols are actually inserted and maintained. After every  $O(n^{1-\epsilon} \log n)$  update operations, the onlyX structure is merged into the original text  $T$  and a new  $T$  is generated. All associated data structures are also rebuilt. Since this construction process could take at most  $O(n \log n)$  time, this cost can be amortized to  $O(n^\epsilon)$  per update. The StaticRankSelect structure on  $T$  takes  $s(n) = n \log |\Sigma| + o(n \log |\Sigma|)$  bits of space. With this frequent rebuilding, all of the other supporting structures take only  $o(n)$  bits of space.

We augment the above two structures with a few additional BitIndel structures. In particular, for each symbol  $s$ , we maintain a bitvector  $I_s$  such that  $I_s[i] = 1$  if and only if the  $i$ th occurrence of  $s$  is stored in the onlyX structure. With the above structures, we quickly describe how to support  $rank_s(i)$  and  $select_s(i)$ .

For  $rank_s(i)$ , we first find  $j = inX.rank_s(i)$ . We then find  $k = inX.rank_x(i)$  and return  $j + onlyX.rank_s(k)$ . For  $select_s(i)$ , we first find whether the  $i$ th occurrence of  $c$  belongs to the inX structure or the onlyX structure. If  $I_s[i] = 0$ , this means that the  $i$ th item is one of the original symbols from  $T$ ; we query  $inX.select_s(j)$  in this case, where  $j = I_s.rank_0(i)$ . Otherwise, we compute  $j = I_s.rank_1(i)$  to translate  $i$  into its corresponding position among new symbols. Then, we compute  $j' = onlyX.select_s(j)$ , its location in  $\hat{T}$  and return  $inX.select_x(j')$ .

Finally, we show how to maintain  $I_s$  during updates. For  $delete(i)$ , compute  $\hat{T}[i] = s$ . We then perform  $I_s.delete(inX.rank_s(i))$ . For  $insert_s(i)$ , after inserting  $s$  in  $\hat{T}$ , we insert it into  $I_s$  by performing  $I_s.insert_1(inX.rank_s(i))$ . Let  $n_x$  be the number of symbols stored in the onlyX structure. We can bound the space for these new BitIndel data structures using RRR [15] and Jensen’s inequality by  $\lceil \log \binom{n'}{n_x} \rceil + o(n') = O(n^{1-\epsilon} \log^2 n) + o(n) = o(n)$  bits of space. Thus, we arrive at the following theorem.

**Theorem 4.** *Given a text  $T$  of length  $n$  drawn from an alphabet  $\Sigma$ , we create a data structure using GMR that takes  $s(n) = n \log |\Sigma| + o(n \log |\Sigma|) + o(n)$  bits of space and supports  $rank_s(i)$ ,  $select_s(i)$ , and  $char(i)$  in  $O(\log \log n + t(n)) = O(\log \log n + \log \log |\Sigma|)$  time and  $insert(i)$  and  $delete(i)$  updates in  $O(n^\epsilon)$  time.  $\square$*

For the special case when  $|\Sigma| = \text{polylog}(n)$ , we may now use [10] as the StaticRankSelect structure, and the Constant Time BitIndel as the BitIndel structure. For the onlyX structure, we can use a similar improvement (using separate select structures for each symbol  $s \in \Sigma$ ) as with BitIndel to achieve  $O(1)$  time queries. The space required is  $o(n)$  if merging is performed every  $O(n^{1-\epsilon})$  update operations. We defer the details of this modification until the full paper. Then, we achieve the following theorem.

**Theorem 5.** *Given a text  $T$  of length  $n$  drawn from an alphabet  $\Sigma$ , with  $|\Sigma| = \text{polylog}(n)$ , we create a data structure using the wavelet tree that takes  $s(n) + o(n) = nH_0 + o(n \log |\Sigma|) + o(n)$  bits of space and supports  $rank_s(i)$ ,  $select_s(i)$ , and  $char(i)$  in  $O(t(n)) = O(1)$  time and  $insert(i)$  and  $delete(i)$  updates in  $O(n^\epsilon)$  time.  $\square$*

We skip the details about the memory allocation issues for our dynamic structures and rebuilding space issues. However, the overhead for these issues can be shown to be  $o(n)$  bits of additional space.

## 4 Dynamizing Ordinal Trees, Labeled Trees, and the XBW Transform

In this section, we describe applications of our BitIndel data structure and our dynamic multi-symbol rank/select data structure to dynamizing ordinal trees, labeled trees, and the XBW transform [2].

**Ordinal Trees.** An *ordinal tree* is a rooted tree where the children are ordered and specified by their rank. An ordinal tree can be represented by the Jacobson’s LOUDS representation [1] using just *rank* and *select*. Thus, we can use our BitIndel data structure to represent any ordinal tree with the following operations:  $v.parent()$ , which returns the parent node of  $v$  in  $T$ ;  $v.child(i)$ , which returns the  $i$ th child node of  $v$ ;  $v.insert(k)$ , which inserts the  $k$ th child of node  $v$ ; and  $v.delete(k)$ , which removes the  $k$ th child of node  $v$ .

**Lemma 7.** *For any ordinal tree  $T$  with  $n$  nodes, there exists a dynamic representation of it that takes at most  $2n + O(n \log \log n / \log n)$  bits of space and supports updates in amortized  $O(n^\epsilon)$  time and navigational queries in  $O(\log \log n)$  time. Alternatively, we can take  $6n + O(n \log \log n / \log n)$  bits of space and support navigational queries in just  $O(1)$  time.  $\square$*

**Labeled Trees, Text Collections, and XBW.** A labeled tree  $T$  is a tree where each of the  $n$  nodes is associated with a label from alphabet  $\Sigma$ . To ease our notation, we will also number our symbols from  $[0, |\Sigma| - 1]$  such that the  $s$ th symbol is also the  $s$ th lexicographically-ordered one. We’ll call this symbol  $s$ . We are interested in constructing a data structure such that it supports the following operations in  $T$ :  $insert(P)$ , which inserts the path  $P$  into  $T$ ;  $v.delete()$ , which removes the root-to- $v$  path for a leaf  $v$ ;  $subpath(P)$ , which finds all occurrences of the path  $P$ ;  $v.parent()$ , which returns the parent node of  $v$  in  $T$ ;  $v.child(i)$ , which returns the  $i$ th child node of  $v$ ; and  $v.child(s)$ , which returns any child node of  $v$  labeled  $s$ .

Ferragina et al. [2] propose an elegant way to solve the static version of this problem by performing an XBW transform on the tree  $T$ , which produces an XBW text  $S$ . They show that storing  $S$  is sufficient to support the desired operations on  $T$  efficiently, namely navigational queries in  $O(\log |\Sigma|)$  time and  $subpath(P)$  queries in  $O(|P| \log |\Sigma|)$  time.

In the dynamic case when we want to support insert or delete of a path of length  $m$ , we observe that either operation corresponds to an update of this XBW text  $S$  at  $m$  positions. Using our dynamic framework, we can then maintain a dynamic version of this text  $S$  and achieve the following result using GMR.

**Theorem 6 (Dynamic XBW).** *For any ordered tree  $T$ , there exists a dynamic succinct representation of it using the XBW transform [2] that takes at most  $s(n) + 2n = n \log |\Sigma| + o(n \log |\Sigma|) + 2n$  bits of space, while supporting navigational queries in  $O(t(n) + \log \log n) = O(\log \log n)$  time. The representation can also answer a  $subpath(P)$  query in  $O(m(t(n) + \log \log n)) = O(m \log \log n)$  time, where  $m$  is the length of path  $P$ . The update operations  $insert(P)$  and  $delete()$*

at node  $u$  for this structure take  $O(n^\epsilon + m(t(n) + \log \log n))$  amortized time, where  $m$  is the length of the path  $P$  being inserted or deleted.  $\square$

**Acknowledgements.** We would like to thank Gonzalo Navarro and S. Muthukrishnan for helpful discussions and reviews of this work.

## References

1. Benoit, D., Demaine, E., Munro, I., Raman, R., Raman, V., Rao, S.: Representing trees of higher degree. *Algorithmica* 43(4), 275–292 (2005)
2. Ferragina, P., Luccio, F., Manzini, G., Muthukrishnan, S.: Structuring labeled trees for optimal succinctness, and beyond. In: *Proceedings of the IEEE Symposium on FOCS*, pp. 184–196 (2005)
3. Ferragina, P., Manzini, G.: Indexing Compressing texts. *JACM*, 52(4), 552–581, (2005)
4. Golynski, A., Munro, J.I., Rao, S.S.: Rank/select operations on large alphabets: a tool for text indexing. In: *SODA*, pp. 368–373 (2006)
5. Grossi, R., Gupta, A., Vitter, J.S.: High-order entropy-compressed text indexes. In: *SODA* (2003)
6. Grossi, R., Vitter, J.S.: Compressed suffix arrays and suffix trees with applications to text indexing and string matching. In: *STOC*, vol. 32 (2000)
7. Hagerup, T., Miltersen, P., Pagh, R.: Deterministic dictionaries. *Journal of Algorithms* 41(1), 353–363 (2001)
8. Hon, W.K., Sadakane, K., Sung, W.K.: Succinct data structures for searchable partial sums. In: *ISAAC*, pp. 505–516 (2003)
9. Jacobson, G.: Succinct static data structures. Technical Report CMU-CS-89-112, Dept. of Computer Science, Carnegie-Mellon University (1989)
10. Navarro, G., Ferragina, P., Manzini, G., Mäkinen, V.: Succinct representation of sequences and full-text indexes. *TALG* (to appear, 2006)
11. Navarro, G., Mäkinen, V.: Compressed full-text indexes. *ACM Computing Surveys* (to appear, 2006)
12. Navarro, G., Mäkinen, V.: Dynamic entropy-compressed sequences and full-text indexes. In: *CPM*, pp. 306–317 (2006)
13. Patrascu, M., Demaine, E.: Logarithmic lower bounds in the cell-probe model. *SIAM Journal on Computing* 35(4), 932–963 (2006)
14. Raman, R., Raman, V., Rao, S.: Succinct dynamic data structures. In: *WADS*, pp. 426–437 (2001)
15. Raman, R., Raman, V., Rao, S.S.: Succinct indexable dictionaries with applications to encoding  $k$ -ary trees and multisets. In: *SODA*, pp. 233–242 (2002)
16. Sadakane, K., Grossi, R.: Squeezing succinct data structures into entropy bounds. In: *SODA*, pp. 1230–1239 (2006)

# In-Place Suffix Sorting

G. Franceschini<sup>1</sup> and S. Muthukrishnan<sup>2</sup>

<sup>1</sup> Department of Computer Science, University of Pisa

francesc@di.unipi.it

<sup>2</sup> Google Inc., NY

muthu@google.com

**Abstract.** Given string  $T = T[1, \dots, n]$ , the *suffix sorting* problem is to lexicographically sort the suffixes  $T[i, \dots, n]$  for all  $i$ . This problem is central to the construction of suffix arrays and trees with many applications in string processing, computational biology and compression. A bottleneck in these applications is the amount of *workspace* needed to perform suffix sorting beyond the space needed to store the input as well as the output. In particular, emphasis is even on the constant  $c$  in the  $O(n) = cn$  space algorithms known for this problem,

Currently the best previous result [5] takes  $O(nv + n \log n)$  time and  $O(n/\sqrt{v})$  extra space, for any  $v \in [1, \sqrt{n}]$  for strings from a general alphabet. We improve this and present the first known in-place suffix sorting algorithm. Our algorithm takes  $O(n \log n)$  time using  $O(1)$  workspace and is optimal in the worst case for the general alphabet.

## 1 Introduction

Given string  $T = T[1, \dots, n]$ , the *suffix sorting* problem is to lexicographically sort the suffixes  $T[i, \dots, n]$  for all  $i$ . Formally, the output is the array  $S$  such that if  $S[j] = k$ ,  $T[k, \dots, n]$  is the  $j$ th smallest suffix [1].

This problem is central to many applications in string processing, computational biology and data compression. For instance, the array  $S$  is in fact the *suffix array* for string  $T$  and is directly applicable to many problems [3]. The classical suffix tree is a compressed trie in which the leaves comprise  $S$ . Finally, the beautiful Burrows-Wheeler transform uses  $S$  to compress  $T$ , and is a popular data compression method.

There are algorithms that will solve this problem using  $O(n)$  *workspace*, i.e., space used in addition to the space needed to store  $T$  and  $S$ . However, in many applications,  $T$  is often very large, for example when  $T$  is a biological sequence, or large corpus. Therefore, for more than a decade, research in this area has been motivated by the fact that even constants in  $O(n)$  matter. For example, the motivation to work with suffix arrays rather than suffix trees arose from decreasing the workspace used from roughly  $11n$  to roughly  $3n$ . Since then the

---

<sup>1</sup> As is standard, we will assume that the end of string is lexicographically smaller than all the other symbols in the string and hence, unequal strings can be compared in a well-defined way.

goal has been to minimize the extra workspace needed. This was explicitly posed as an open problem in [2].

Currently the best previous result [5] takes  $O(nv + n \log n)$  time and  $O(n/\sqrt{v})$  extra space, for any  $v \in [1, \sqrt{n}]$ . Here we assume the general alphabet model in which the string elements can be compared pairwise.<sup>2</sup> This has a variety of tradeoffs: one is  $O(n \log n)$  time and  $O(n)$  space, and the other is  $O(n^{3/2})$  time and  $O(n^{1/4})$  space, depending on  $v$ .

Our main result is a substantial improvement over the above. In particular, we present the first known in-place suffix sorting algorithm, that is, our algorithm uses only  $O(1)$  workspace. The running time of our algorithm is  $O(n \log n)$  which is optimal in the general alphabet model since even sorting the  $n$  characters will take that much time in the worst case. Formally, we prove:

**Theorem 1.** *The suffixes of a text  $T$  of  $n$  characters drawn from a general alphabet  $\Sigma$  can be sorted in  $O(n \log n)$  time using  $O(1)$  locations besides the ones for  $T$  and the suffix array  $S$  of  $T$ .*

## 2 Preliminaries

We are given a text  $T$  of  $n$  characters drawn from a general alphabet  $\Sigma$  and an array  $S$  of  $n$  integers of  $\lceil \log n \rceil$  bits each. A total order relation  $\leq$  is defined on  $\Sigma$ , the characters are considered atomic (no bit manipulation, hashing or word operations) and the only operations allowed on them are comparisons (w.r.t.  $\leq$ ). Let  $T_i$  be the suffix of  $T$  starting with the character  $T[i]$  (i.e.  $T_i = T[i]T[i+1] \cdots T[n]$ ) and let integer  $i$  be the *suffix index* of  $T_i$ . The objective is to sort lexicographically the  $n$  suffixes of  $T$ . The result, consisting of the  $n$  suffix indices permuted according to the lexicographical order of the suffixes, is to be stored in  $S$ . Apart from accessing  $T$  (readonly) and  $S$ , we are allowed to use *only*  $O(1)$  integers of  $\lceil \log n \rceil$  bits each to carry out the computation.

In the following we denote with  $\prec$  the lexicographical order relation. For any suffix  $T_i$ , we refer to  $T_{i-1}$  ( $T_{i+1}$ ) as the *text-predecessor* (*text-successor*) of  $T_i$ . The terms *sequence* and *subarray* will have slightly different meanings. Even though they are both composed by contiguous elements of an array, a subarray is intended to be just *a static portion* of an array while sequences are *dynamic* and can be moved, exchanged, permuted etc. For any string  $A$  we denote with  $A[i \dots j]$  the contiguous substring going from the  $i$ -th position to the  $j$ -th position of  $A$ . We extend the same notation to arrays and sequences.

### 2.1 A Space Consuming Approach

The strategy for sorting suffixes in our solution is based on the simple and elegant approach by Ko and Aluru in [6]. Even though their technique was originally used for the case where  $\Sigma = \{1, \dots, n\}$ , it can be extended to the comparison

<sup>2</sup> The bounds in [5] are for strings with integer alphabet. The bound we have quoted is the best possible time bound they can achieve in the general alphabet model.

model. The result is a suffix sorting algorithm with an optimal  $O(n \log n)$  time complexity but requiring  $O(n)$  auxiliary locations in addition to  $S$ .

Let us recall Ko and Aluru's approach. The suffixes of  $T$  are classified as follows: a suffix  $T_i$  is an  $\alpha$ -suffix (a  $\beta$ -suffix) if  $T_i \prec T_{i+1}$  ( $T_{i+1} \prec T_i$ ), that is if it is less (greater) than its text-successor w.r.t. the lexicographical order ( $T_n$  is classified as a  $\beta$ -suffix by convention). This classification has the following main property: for any  $\alpha$ -suffix  $T_i$  and any  $\beta$ -suffix  $T_j$ , if  $T[i] = T[j]$  then  $T_j \prec T_i$ .

Let us assume without loss of generality that the  $\alpha$ -suffixes of  $T$  are fewer in number than the  $\beta$ -suffixes. An  $\alpha$ -substring of  $T$  is a substring  $A_i = T[i]T[i+1] \cdots T[i']$  such that (i) both  $T_i$  and  $T_{i'}$  are  $\alpha$ -suffixes and (ii)  $T_j$  is a  $\beta$ -suffix, for any  $i < j < i'$ . We need to sort the  $\alpha$ -substring of  $T$  according to a variation of the lexicographical order relation, the *in-lexicographical order*, from which it differs in only one case: if a string  $s$  is a prefix of another string  $s'$ , then  $s$  follows  $s'$ . For any multiset  $\mathcal{M}$  (i.e. a collection allowing duplicates), for any total order  $<$  defined on  $\mathcal{M}$  and for any  $o \in \mathcal{M}$ , the *bucket number of  $o$*  is the rank of  $o$  according to  $<$  in the set  $\mathcal{S}_{\mathcal{M}}$  obtained from  $\mathcal{M}$  by removing all the duplicates.

The Ko and Aluru's approach proceeds with the following three main steps.

*First.* We sort *in-lexicographically* the  $\alpha$ -substrings of  $T$ .

*Second.* We build a string  $T'$  from  $T$  by replacing any  $\alpha$ -substring with its bucket number (according to the *in-lexicographical order*). Then, we sort the suffixes of  $T'$  recursively, obtaining the corresponding array  $S'$ . Because of the main property of the  $\alpha$ - $\beta$  classification and by the definition of the *in-lexicographical order*, sorting the suffixes of  $T'$  is equivalent to sorting the  $\alpha$ -suffixes of  $T$ .

*Third.* We distribute the suffixes into temporary buckets according to their first character. By the main property of the  $\alpha$ - $\beta$  classification we know that any  $\alpha$ -suffix is greater than any  $\beta$ -suffix belonging to the same bucket. Therefore, for any bucket, we move all its  $\alpha$ -suffixes to its right end and we dispose them in lexicographical order (known since the second step). Then, we move the suffixes from the temporary buckets to  $S$  (in the same order we find them in the temporary buckets). Finally, we take care of the  $\beta$ -suffixes. We scan  $S$  from left to right. Let  $T_i$  be the currently examined suffix. If the text-predecessor of  $T_i$  is an  $\alpha$ -suffix, we ignore it (it is already in its final position in  $S$ ). Otherwise, if  $T_{i-1}$  is a  $\beta$ -suffix, we exchange  $T_{i-1}$  with the leftmost  $\beta$ -suffix  $T_j$  having the same first character as  $T_{i-1}$  and not yet in its final position (if any). After the scanning process, the suffixes are in lexicographical order.

## 2.2 Obstacles

Before we proceed with the description of our algorithm, let us briefly consider some of the obstacles that we will have to overcome.

### 2.2.1 Input Partitioning and Simulated Resources

A common approach for attacking space complexity problems consists of the following phases. First, the input set is partitioned into two disjoint subsets. Then, the problem is solved for the first subset using the second subset to simulate additional space resources. Usually these simulated resources are implemented



by permuting the elements in the second subset in order to encode data or, if the model allows it, by compressing them in order to free some bits temporarily. After the problem has been solved for the first subset, the approach is applied recursively on the second one. Finally, the partial solutions for the two subsets are merged into one.

Unfortunately, this basic approach cannot be easily extended to the suffix sorting problem. This is due to the well-known fact that the suffixes of a sequence cannot be just partitioned into generic subsets to be sorted separately and then merged efficiently. Only few specific types of partitionings are known to have this property and either they exploit some cyclic scheme (e.g. [4]), thus being too rigid for our purposes, or they need to be explicitly represented (e.g. [6]) thereby increasing the auxiliary memory requirements.

### 2.2.2 Auxiliary Information Needed in Ko and Aluru's Approach

Not only is the  $\alpha$ - $\beta$  partitioning unsuitable, but it also claims auxiliary resources. Clearly, in the first and third main steps of the Ko and Aluru's approach, we need to be able to establish whether a particular suffix is an  $\alpha$ -suffix or a  $\beta$ -suffix. The number of bits needed to represent the  $\alpha$ - $\beta$  partitioning can be reduced to  $n/c$ , for a suitably large integer constant  $c$ . We will employ various encoding schemes to maintain this information implicitly during the phases of the computation.

Let us consider the final scanning process of the third main step. For any suffix  $T_i$  considered, the positions in  $S$  of  $T_{i-1}$  and  $T_j$  (the leftmost  $\beta$ -suffix such that  $T_j[1] = T_{i-1}[1]$  and not yet in its final position) must be retrieved efficiently. To this end, the algorithm in [6] explicitly stores and maintains the inverse array of  $S$ . Unlike the case of the  $\alpha$ - $\beta$  partitioning, it is clearly impossible to encode implicitly the necessary  $n \lceil \log n \rceil$  bits. Therefore, we will devise an "on-the-fly" approach to the scanning process that will require neither the exchange step of  $T_{i-1}$  and  $T_j$  nor the use of any implicit encoding scheme.

## 3 Our Algorithm

In this section we present our optimal in-place suffix sorting algorithm for generic alphabets. We will assume without loss of generality that the  $\alpha$ -suffixes of  $T$  are fewer in number than the  $\beta$ -suffixes (the other case is symmetric).

**The  $\alpha$ - $\beta$  table.** In [6] the information for classifying  $\alpha$  and  $\beta$ -suffixes is calculated in linear time with a left to right scan of  $T$ . It is easy to see how this information can be gathered in linear time also by a right to left scan: by convention  $T_n$  is classified as  $\alpha$ -suffix and  $T_{n-1}$  can be classified by comparing  $T[n]$  and  $T[n-1]$ ; for any  $i < n-1$ , let us assume inductively that the classification of  $T_{i+1}$  is known, if  $T[i] \neq T[i+1]$  then  $T_i$  is classified by the result of the comparison between  $T[i]$  and  $T[i+1]$ , otherwise  $T_i$  is of the same type of  $T_{i+1}$ . Therefore, to be able to classify any suffix  $T_i$  in  $O(1)$  time there is no need to store a table with  $n$  entries of one bit each. For any integer constant  $c$ , we can use a table of  $n/c$  bits whose  $j$ -th entry represents the classification of  $T_{cj}$ . Any suffix  $T_i$  can be classified in  $O(c)$  time: if the substrings

$T_i[1 \dots c - i \bmod c]$  and  $T_{i+1}[1 \dots c - i \bmod c]$  differ then  $T_i$  is classified by the result of their lexicographical comparison, otherwise  $T_i$  is of the same type of  $T_{i+c-i \bmod c}$  (whose classification is in the  $\frac{i+c-i \bmod c}{c}$ -th entry of the table).

We will refer to this smaller table as the  $\alpha$ - $\beta$  table. We will not be able to keep the  $\alpha$ - $\beta$  table explicitly stored or implicitly encoded all the time. Its information will be lost and recalculated multiple times during any execution.

### 3.1 Sorting the $\alpha$ -Suffixes

In this section we show how to sort the  $\alpha$ -suffixes of  $T$ . We have four phases.

#### 3.1.1 First Phase

We compute the  $\alpha$ - $\beta$  table and we store it in  $S[1 \dots n/c]$  (that is, simply using one entry of  $S$  for any entry of the table). Let  $n_\alpha$  be the number of  $\alpha$ -suffixes. While we are scanning  $T$  to compute the  $\alpha$ - $\beta$  table we also find the  $\alpha$ -substrings and we store the  $n_\alpha$  pointers to them in  $S[n - n_\alpha + 1, \dots, n]$ . Since  $n_\alpha \leq n/2$ , in the following phases we can exploit  $n(1/2 - 1/c)$  free locations in the first half of  $S$  (as we have seen, we can choose the constant  $c$ , defining the size of the  $\alpha$ - $\beta$  table, as large as we want). Let  $F$  denote the subarray of  $S$  containing the free locations.

#### 3.1.2 Second Phase

We divide the pointers to the  $\alpha$ -substrings into  $d$  groups  $G_1, \dots, G_d$  of  $n_\alpha/d$  contiguous pointers each, for a suitable constant  $d$ . Then, we sort each group  $i$ -lexicographically using the locations in the subarray  $F$  (and the  $\alpha$ - $\beta$  table to recognize the last position of any  $\alpha$ -substring). As we can choose  $d$  as large as we want and given that the total length of the  $\alpha$ -substrings is  $O(n)$ , each group can be sorted in  $O(n \log n)$  time with any optimal string sorting algorithm operating in linear space (linear w.r.t. the size of the group).

Now that the groups of  $\alpha$ -substrings are  $i$ -lexicographically sorted we merge them. We first merge the first group with the second one, then the resulting sequence is merged with the third group and so forth. For any  $i$ , the  $i$ -th single binary merging step is performed with the help of the locations in  $F$  in the following way. Let  $G$  be the sequence to be merged with the  $i$ -th group  $G_i$  and let us assume that  $|G| + |G_i| > |F|$  (at a certain point this will happen, since  $n_\alpha > |F|$ ). Using the  $\alpha$ - $\beta$  table to recognize the ends of the  $\alpha$ -substrings, we initially proceed like in a normal merging (we compare  $i$ -lexicographically the  $\alpha$ -substrings pointed by  $G[1]$  and  $G_i[1]$  and move a pointer to  $F[1]$  and so forth). Since  $|G| + |G_i| > |F|$ , after  $|F|$  of these single string comparison steps  $F$  becomes full. At that point we slide what is left of  $G$  to the right so that it becomes adjacent with what is left of  $G_i$ . After the sliding we resume the merging but now we move the pointers to the subarray of  $|F|$  positions that is right after  $F$  and that has become free after the sliding. We proceed in this fashion until  $G$  or  $G_i$  is empty. At that point we compact the sorted pointers with what is left of  $G$  or  $G_i$ . Finally, we slide the resulting sequence to the right to be adjacent with  $G_{i+1}$  (in case we need to).

### 3.1.3 Third Phase

In this phase we build a sequence  $T'$  of  $n_\alpha$  integers in the range  $[1, n_\alpha]$  using the bucket numbers of the  $\alpha$ -substrings (see Section 2.1). After the second phase the  $\alpha$ -substrings are in *in*-lexicographical order and the pointers to them are permuted accordingly. Let us denote with  $P$  the subarray  $S[n - n_\alpha + 1 \dots n]$  where the pointers are stored.

We start by modifying the allocation scheme for the  $\alpha$ - $\beta$  table. Up until now the  $n/c$  entries have been stored in the first  $n/c$  locations of  $S$ . We now allocate the  $\alpha$ - $\beta$  table so that the  $i$ -th entry is stored in the most significant bit of the  $2i$ -th location of  $S$ , for any  $1 \leq i \leq n/c$ . Since we can choose  $c$  to be as large as we want, the  $n_\alpha$  pointers residing in  $P$  will not be affected by the change.

Then, we associate to any pointer the bucket number of the  $\alpha$ -substring it points to in the following way. We scan  $P$  from left to right. Let two auxiliary variables  $j$  and  $p$  be initially set to 1 and  $P[1]$ , respectively. In the first step of the scanning we set  $S[1]$  and  $S[2]$  to  $P[1]$  and 1, respectively. Let us consider the generic  $i$ -th step of the scanning, for  $i > 1$ .

1. We compare *in*-lexicographically the  $\alpha$ -substrings pointed by  $p$  and  $P[i]$  (using the  $\alpha$ - $\beta$  table to recognize the last positions of the two  $\alpha$ -substrings).
2. If they are different we increment  $j$  by one and we set  $p$  to  $P[i]$ .
3. In any case, we set  $S[2i - 1]$  and  $S[2i]$  to  $P[i]$  and  $j$ , respectively and we continue the scanning.

As we said, the scanning process depends on the  $\alpha$ - $\beta$  table for the substring comparisons. It might seem that writing the current value of  $j$  in the locations of  $S$  with even indices would destroy the  $\alpha$ - $\beta$  table. That is not the case. After the scanning process, the first  $2n_\alpha$  locations of  $S$  contains  $n_\alpha$  pairs  $\langle p_i, b_i \rangle$ , where  $p_i$  is a pointer to the  $i$ -th  $\alpha$ -substring (in *in*-lexicographical order) and  $b_i$  is the bucket number of it. Since  $n_\alpha \leq n/2$ , the bits necessary to represent a bucket number for an  $\alpha$ -substrings are no more than  $\lceil \log(n/2) \rceil$  (and the locations of  $S$  have  $\lceil \log n \rceil$  bits each). Therefore, the  $n/c$  entries of the  $\alpha$ - $\beta$  table and the bucket numbers of the first  $n/c$  pair  $\langle p_i, b_i \rangle$  can coexist without problems.

After the scanning process we proceed to sort the  $n_\alpha$  pairs  $\langle p_i, b_i \rangle$  according to their first members (i.e. the pointers are the sorting keys). Since there can be as many as  $n/2$   $\alpha$ -substrings, at the worst case we have that all the locations of  $S$  are occupied by the sequence of pairs. Therefore, for sorting the pairs we use mergesort together with an in-place, linear time merging like [8]. When the pairs are sorted, we scan them one last time to remove all the pointers, ending up with the wanted sequence  $T'$  stored in the first  $n_\alpha$  locations of  $S$ .

### 3.1.4 Fourth Phase

We start by applying our in-place algorithm recursively to the sequence  $T'$  stored in  $S[1 \dots n_\alpha]$ . At the worst case  $|T'| = n_\alpha$  can be equal to  $n/2$  and the space used by the algorithm to return the  $n_\alpha$  sorted suffixes of  $T'$  is the subarray  $S' = S[n - n_\alpha + 1 \dots n]$ . Concerning the use of recursion, if before any recursive call we were to store explicitly  $O(1)$  integer values (e.g. the value of  $n_\alpha$ ), we would end up using  $O(\log n)$  auxiliary locations ( $O(\log n)$  nested recursive calls). There

are many solutions to this problem. We use the  $n$  most significant bits of  $S$  (the most significant bit in each of the  $n$  entries in  $S$ ) to store the  $O(1)$  integers we need for any recursive call. That is possible because starting from the first recursive call the alphabet of the text is not  $\Sigma$  anymore but  $\{1, 2, \dots, n/2\}$  and the size of the input becomes at most  $n/2$ . Therefore, the  $n$  most significant bits of  $S$  are untouched during all the recursive calls.

After the recursive call, we have  $n_\alpha$  integers of  $\lceil \log(n/2) \rceil$  bits each stored in the subarray  $S' = S[n - n_\alpha + 1 \dots n]$ . They are the suffix indices of the sequence  $T'$  stored in the subarray  $S[1 \dots n_\alpha]$  and they are permuted according to the lexicographical order of the suffixes of  $T'$ .

Finally, to obtain the lexicographical order of the  $\alpha$ -suffixes of  $T$  we proceed as follows. First, scanning  $T$  as we do to compute the  $\alpha$ - $\beta$  table, we recover the indices of the  $\alpha$ -suffixes and we store them in  $T'$  (we do not need the data in  $T'$  anymore). Then, for any  $1 \leq i \leq n_\alpha$ , we set  $S'[i] = T'[S'[i]]$ .

### 3.2 Sorting the Suffixes

In this section we show how to sort the suffixes of  $T$  provided the  $\alpha$ -suffixes are already sorted. Let us assume that the suffix indices for the  $\alpha$ -suffixes are stored in  $S[n - n_\alpha + 1 \dots n]$ . Before we start let us recall that any two adjacent sequences  $U$  and  $V$ , possibly with different sizes, can be exchanged in-place and in linear time with three sequence reversals, since  $UV = (V^R U^R)^R$ .

We have six phases.

#### 3.2.1 First Phase

With a process analogous to the one used to compute the  $\alpha$ - $\beta$  table (which we do not have at our disposal at this time), we scan  $T$ , recover the  $n - n_\alpha$  suffix indices of the  $\beta$ -suffixes and store them in  $S[1 \dots n - n_\alpha]$ .

Then, we sort the pointers stored in  $S[1 \dots n - n_\alpha]$  according to the first character of their respective  $\beta$ -suffixes (i.e. the sorting key for  $S[i]$  is  $T[S[i]]$ ). We use mergesort together with the in-place, linear time merging in [8].

#### 3.2.2 Second Phase

Let us denote  $S[1 \dots n - n_\alpha]$  and  $S[n - n_\alpha + 1 \dots n]$  by  $S_\beta$  and  $S_\alpha$ , respectively. After the first phase the pointers in  $S_\beta$  are sorted according to the first character of their suffixes and so are the ones in  $S_\alpha$ .

Scanning  $S_\beta$ , we find the *rightmost location*  $j_\beta$  such that the following hold:

- (i)  $S_\beta[j_\beta]$  is the leftmost pointer in  $S_\beta$  whose  $\beta$ -suffix has  $T[S_\beta[j_\beta]]$  as first character.
- (ii)  $n - n_\alpha - j_\beta + 1 \geq 2n/c$ , that is, the number of pointers in the subarray  $S_\beta[j_\beta \dots n - n_\alpha]$  is at least two times the number of entries of the  $\alpha$ - $\beta$  table.

The reason for this choice will be clear at the end of this phase.

Then, we find the leftmost position  $j_\alpha$  in  $S_\alpha$  such that  $T[S_\alpha[j_\alpha]] \geq T[S_\beta[j_\beta]]$  (a binary search). Let us consider the pointers in  $S$  as belonging to four sequences  $B', B'', A'$  and  $A''$ , corresponding to the pointers in  $S_\beta[1 \dots j_\beta - 1]$ ,

$S_\beta[j_\beta \dots n - n_\alpha]$ ,  $S_\alpha[1 \dots j_\alpha - 1]$  and  $S_\alpha[j_\alpha \dots n_\alpha]$ , respectively. We exchange the places of the sequences  $B''$  and  $A'$ .

For the sake of presentation, let us assume that the choice of  $j_\beta$  is *balanced* that is condition (ii) holds for subarray  $S_\beta[1 \dots j_\beta - 1]$  too. The other case requires only minor, mainly technical, modifications to the final phases and we will discuss it in the full version of this paper.

In the final step of this phase, we calculate the  $\alpha$ - $\beta$  table by scanning  $T$  and while we are doing so we encode the table in  $B''$  in the following way: if the  $i$ -th entry of the table is 0 (1) we exchange positions of the pointers  $B''[2i - 1]$  and  $B''[2i]$  so that they are in ascending (descending) order. It is important to point out that in this case we mean the relative order of *the two pointers themselves*, seen as simple integer numbers, and not the relative order of the suffixes pointed by them. Clearly, any entry of the table can be decoded in  $O(1)$  time.

This basic encoding technique is known as odd-even encoding ([7]). Its main advantage w.r.t. other, more sophisticated, encoding techniques is its extreme simplicity. Its main drawback is that it introduces an  $\omega(1)$  overhead if used to encode/decode a table with entries of  $\omega(1)$  bits. Since we will use it only to encode the  $\alpha$ - $\beta$  table, the overhead will not be a problem.

At the end of the second phase the pointers in the  $S$  are divided into the four contiguous sequences  $B'A'B''A''$  and the  $\alpha$ - $\beta$  table is implicitly encoded in  $B''$ . For the rest of the paper let us denote with  $S_L$  and  $S_R$  the subarrays  $S[1 \dots |B'| + |A'|]$  and  $S[|B'| + |A'| + 1 \dots n]$ , respectively (i.e. the two subarrays of  $S$  containing all the pointers in  $B'A'$  and in  $B''A''$ ).

### 3.2.3 Third Phase

We start by merging *stably* the pointers in  $B'$  and  $A'$  according to the first character of their suffixes. So, the sorting key for pointer  $B'[i]$  is  $T[B'[i]]$  and the relative order of pointers with equal keys is maintained. For this process we use the stable, in-place, linear time merging in [8].

After the merging, the pointers in  $S_L$  are contained into  $m$  contiguous sequences  $C_1 C_2 \dots C_m$  where  $m$  is the cardinality of the set  $\{T[S_L[i]] \mid 1 \leq i \leq |S_L|\}$  and for any  $j$  and  $p', p'' \in C_j$ ,  $T[p'] = T[p'']$ . Let us recall that  $A'$  contained the pointers to the  $|A'|$  lexicographically smallest  $\alpha$ -suffixes and they were already in lexicographical order. Therefore, since we merged  $B'$  and  $A'$  stably, we know that any sequence  $C_j$  is composed by two contiguous subsequences,  $C_j^\beta$  followed by  $C_j^\alpha$ , such that (i)  $C_j^\beta$  contains only pointers to  $\beta$ -suffixes and (ii)  $C_j^\alpha$  contains only pointers to  $\alpha$ -suffixes and they are already in lexicographical order.

Now we need to gather some elements from some of the sequences  $C_i$  into a sequence  $E$ . We process each sequence  $C_i$  starting from  $C_1$ . Initially  $E$  is void. Let us assume that we have reached the location  $c_j$  in  $S_L$  where the generic subsequence  $C_j$  begins and let us assume that at this time  $E$  is located right before  $C_j$ . We process  $C_j$  with the following steps.

1. We find the ending locations of  $C_j^\beta$  and  $C_j^\alpha$ , using the  $\alpha$ - $\beta$  table encoded in sequence  $B''$  of  $S_R$  (e.g. with a linear scan of  $C_j$ ).

2. If  $C_j^\beta$  contains at least two pointers we proceed as follows.
  - (a) We “mark” the second location of  $C_j^\beta$  by setting  $C_j^\beta[2] = S_L[1]$ . We can employ  $S_L[1]$  as a “special value” since, by construction, it has to point to an  $\alpha$ -suffix and so it is not among the values affected by this process.
  - (b) We enlarge  $E$  by one element at its right end (thus including the first element of  $C_j^\beta$  and shrinking  $C_j$  by one element at its left end).
3. We move  $E$  (which may have been enlarged by one element in step 2) past  $C_j$  in the following way. If  $|E| \leq |C_j|$ , we simply exchange them. Otherwise, if  $|E| > |C_j|$ , we exchange the first  $|C_j|$  elements of  $E$  with  $C_j$ , thus “rotating”  $E$ . (If we just exchanged  $E$  and  $C_j$  all the times, the cost of the whole scanning process would be  $O(n^2)$ ).

After this first scanning process,  $E$  resides at the right end of  $S_L$ . Moreover,  $|E|$  sequences among  $C_1C_2 \dots C_m$  had their first pointer to a  $\beta$ -suffix moved at the right end of  $S_L$  and their second pointer to a  $\beta$ -suffix overwritten with the “special value”  $S_L[1]$ .

We proceed with a second scanning of the sequences  $C_1C_2 \dots C_m$  from left to right. This time we remove the “special values”  $S_L[1]$  in every location we find it (except the first location of  $S_L$ ) by compacting the sequences toward left. With this process  $|E|$  locations are freed right before sequence  $E$ . (Clearly, any sequence  $C_i$  that before the two scanings had  $|C_i| = 2$  and contained only pointers to  $\beta$ -suffixes has now disappeared.)

Finally, we create a “directory” in the last  $2|E|$  locations of  $S_L$  (the second scanning has freed the  $|E|$  locations before sequence  $E$ ). Let us denote with  $G_L$  and  $D_L$  the subarrays with the first  $|S_L| - 2|E|$  and the last  $2|E|$  locations of  $S_L$ , respectively. We proceed with the following steps.

1. We use mergesort with the in-place, linear time binary merging in [8] to sort the elements of  $E$ , for any  $1 \leq i \leq |E|$  we use  $T[E[i]]$  as sorting key.
2. We “spread”  $E$  through  $D_L$ , that is we move  $E[1]$  to  $D_L[1]$ ,  $E[2]$  to  $D_L[3]$ ,  $\dots$ ,  $E[i]$  to  $D_L[2i - 1]$  etc.
3. For any  $1 \leq i \leq |E|$ . We do a binary search for the character  $t_i = T[D_L[2i - 1]]$  in  $G_L$  using the character  $T[G_L[l]]$  as key for the  $l$ -th entry of  $G_L$ . The search returns the leftmost position  $p_i$  in  $G_L$  where  $t_i$  could be inserted to maintain a sorted sequence. We set  $D_L[2i] = p_i$ .

### 3.2.4 Fourth Phase

In this phase we finalize the sorting of the  $|S_L|$  lexicographically smallest suffixes of  $T$  (and their pointers will be stored in  $S_L$ ).

We start the phase by scanning  $G_L$  from left to right. Let us remark that, by construction,  $G_L[1]$  contains a pointer to an  $\alpha$ -suffix, the lexicographically smallest suffix of  $T$ . For the generic  $i$ -th location of  $G_L$  we perform two main steps. First we will give a fairly detailed and formal description of these two steps and then we will give a more intuitive explanation of them.

1. We do a binary search for the character  $T[G_L[i] - 1]$  (i.e. the first character of the text-predecessor of the suffix pointed by  $G_L[i]$ ) in the directory in  $D_L$ . The binary search is done on the odd locations of  $D_L$  (the ones with pointers to  $T$ ) by considering the character  $T[D_L[2l - 1]]$  as the key for the  $l$ -th odd location of  $D_L$ .
2. If the binary search succeeded, let  $j$  be the (odd) index in  $D_L$  such that  $T[G_L[i] - 1] = T[D_L[j]]$ . Let  $p_j$  be the inward pointer stored in  $D_L[j + 1]$ . We use  $p_j$  to place the outward pointer to the text-predecessor of the suffix pointed by  $G_L[i]$  in a position in  $S_L$  (not only in  $G_L$ ). We have three cases:
  - (a) If  $T[G_L[i] - 1] = T[G_L[p_j]]$  and  $G_L[p_j]$  is a  $\beta$ -suffix (we verify this using the  $\alpha$ - $\beta$  table encoded in sequence  $B''$  of  $S_R$ ), we set  $G_L[p_j]$  to  $G_L[i] - 1$  and we increment  $D_L[j + 1]$  (i.e.  $p_j$ ) by one.
  - (b) If  $G_L[p_j]$  is an  $\alpha$ -suffix or  $T[G_L[i] - 1] \neq T[G_L[p_j]]$ , we set  $D_L[j]$  to  $G_L[i] - 1$  and  $D_L[j + 1]$  (i.e.  $p_j$ ) to  $|G_L| + 1$ .
  - (c) If  $p_j > |G_L|$ , we set  $D_L[j + 1] = G_L[i] - 1$ .

And now for the intuitive explanation. As we anticipated in Section 2.2.2, the scanning process of the third main step (see Section 2.1) needs to maintain the inverse array of  $S$  in order to find the correct position for the text-predecessor of  $T_i$  in its bucket. Obviously we cannot encode that much information and so we develop an “on-the-fly” approach. The directory in  $D_L$  is used to find an inward (i.e. toward  $S$  itself and not  $T$ ) pointer to a location in one of the  $C_k$  sequences (which represent the buckets in our algorithm). So the first step simply uses the directory to find the inward pointer. Unfortunately the directory itself is stealing positions from the sequences  $C_k$  that sooner or later in the scanning will be needed to place some outward pointer (i.e. toward  $T$ ). For any sequence  $C_k$  that has a “representative”, that is a pair  $\langle p_{out}, p_{in} \rangle$ , in the directory (the sequences without representatives are already in lexicographical order as they contain only one  $\beta$ -suffix) we have three cases to solve.

In the first case (corresponding to step 2a), there is still space in  $C_k$ , that is the two lexicographically largest  $\beta$ -suffixes belonging to  $C_k$  have not yet been considered by the scanning (these are the suffixes of  $C_k$  whose space in  $S$  is stolen by the directory in  $D_L$ ). In this case the inward pointer we found in the directory guides us directly to the right position in  $C_k$ .

In the second case (corresponding to step 2b) the space in  $C_k$  is full and the suffix whose pointer we are trying to place is the lexicographically second largest  $\beta$ -suffix belonging to  $C_k$ . To solve the problem, we place its outward pointer in the first location of the pair  $\langle p_{out}, p_{in} \rangle$  in  $D_L$  corresponding to the bucket  $C_k$ . This overwrites the outward pointer  $p_{out}$  that we use in the binary search but this is not a problem, as the old one and the new one belong to the same bucket. The only problem is that we need to be able to distinguish this second case from the third case, when the last  $\beta$ -suffix belonging to  $C_k$  will be considered in the scan. To do so we set  $p_{in}$  to a value that cannot be a valid inward pointer for the phase (e.g.  $|G_L| + 1$ ).

In the third case (corresponding to step 2c) the space in  $C_k$  is full, and the pointer to the lexicographically second largest  $\beta$ -suffix belonging to  $C_k$  has been

placed in the first location of the pair  $\langle p_{out}, p_{in} \rangle$  in  $D_L$  corresponding to  $C_k$ . When the largest  $\beta$ -suffix of  $C_k$  is finally considered in the scanning, we are able to recognize this case since the value of the  $p_{in}$  is an invalid one ( $|G_L| + 1$ ). Then, we set  $p_{in}$  to be the pointer to the largest  $\beta$ -suffix of  $C_k$  and, for what concerns  $C_k$ , we are done.

After the scanning process, the pointers in  $G_L$  are permuted according to the lexicographical order of their corresponding suffixes. The same holds for  $D_L$ . Moreover, for any  $1 \leq j \leq |D_L|/2$ , the suffixes pointed by  $D_L[2j - 1]$  and  $D_L[2j]$  (a) have the same first character, (b) are both  $\beta$ -suffixes and (c) they are the second largest and largest  $\beta$ -suffixes among the ones with their same first character, respectively. Knowing these three facts, we can merge the pointers in  $G_L$  and  $D_L$  in two steps.

1. We merge  $G_L$  and  $D_L$  *stably* with the in-place, linear time binary merging in [8] using  $T[G_L[i]]$  and  $T[D_L[j]]$  as merging keys for any  $i, j$ .
2. Since we merged  $G_L$  and  $D_L$  *stably*, after step [1] any pair of  $\beta$ -suffixes with the same first character whose two pointers were previously stored in  $D_L$ , now *follow* immediately (instead of preceding) the  $\alpha$ -suffixes with their same first character (if any). A simple right to left scan of  $S_L$  (using the encoded  $\alpha$ - $\beta$  table) is sufficient to correct this problem in linear time.

### 3.2.5 Fifth Phase

We start the phase by sorting the pointers in the sequence  $B''$  of  $S_R$  according to the first character of their corresponding suffixes. The pointers in  $B''$  were already in this order before we perturbed them to encode the  $\alpha$ - $\beta$  table. Hence, we can just scan  $B''$  and exchange any two pointers  $B[i]$  and  $B[i + 1]$  such that  $T[B[i]] > T[B[i + 1]]$  (since the pointers in  $B''$  are for  $\beta$ -suffixes, we do not need to care about their relative order when  $T[B[i]] = T[B[i + 1]]$ ).

After that, we process  $S_R$  in the same way we processed  $S_L$  in the third phase (Section 3.2.3). Since  $\alpha$ - $\beta$  table was used in that process, we need to encode it somewhere in  $S_L$ .

Unfortunately, we cannot use the plain odd-even encoding on the suffix indices in  $S_L$  because they are now in lexicographical order w.r.t. to the suffixes they point to. If we encoded each bit of the  $\alpha$ - $\beta$  table by exchanging positions of two *consecutive* pointers  $S_L[i]$  and  $S_L[i + 1]$  (like we did with sequence  $B''$  in the second phase, Section 3.2.2), then, after we are done using the encoded table, in order to recover the lexicographical order in  $S_L$  we would have to compare  $n/c$  pairs of consecutive suffixes. At the worst case that can require  $O(n^2)$  time.

Instead, we exploit the fact that pointers in  $S_L$  are in lexicographical order w.r.t. to their suffixes in the following way. Let  $\text{Tab}'_L$  and  $\text{Tab}''_L$  be the subarrays  $S_L[1 \dots n/c]$  and  $S_L[n/c + 1 \dots 2n/c]$ , respectively. We encode a smaller  $\alpha$ - $\beta$  table with only  $n/2c$  entries in  $\text{Tab}''_L$  as follows ( $\text{Tab}'_L$  will be used in the sixth phase): if the  $i$ -th entry of the table is 0 (1) we exchange positions of the pointers  $\text{Tab}''_L[i]$  and  $\text{Tab}''_L[|\text{Tab}''_L| - i + 1]$  so that they are in ascending (descending) order (as before, in this case we mean the relative order of *the two pointers themselves*, seen as simple integer numbers).



Since the pointers in  $\text{Tab}_L''$  were in lexicographical sorted order before the pair swapping, the recovering of the order of all the pairs of  $\text{Tab}_L''$  can be achieved in  $O(n)$  time in the following way. Let us denote with  $p_1, p_2, \dots, p_t$  be the pointers in  $\text{Tab}_L''$  after the encoding (where  $t = |\text{Tab}_L''|$ ). We start lexicographically comparing the suffixes  $T_{p_1}$  and  $T_{p_t}$ . When we find the mismatch, let it be at the  $h$ -th pair of characters, we exchange  $p_1$  and  $p_t$  accordingly. Then we proceed by comparing  $T_{p_2}$  and  $T_{p_{t-1}}$  but we do not start comparing them by their first characters. Since  $\text{Tab}_L''$  was in lexicographical sorted order, we know that the first  $h - 1$  characters must be equal. Hence, we start comparing  $T_{p_2}$  and  $T_{p_{t-1}}$  starting from their  $h$ -th characters. We proceed in this fashion until the last pair has been dealt with. Clearly this process takes  $O(n)$  time at the worst case.

### 3.2.6 Sixth Phase

In this phase we finalize the sorting of the remaining  $|S_R|$  lexicographically largest suffixes of  $T$  (and their pointers will be stored in  $S_R$ ).

The final sorting process for  $S_R$  is the same we used in the fourth phase (Section 3.2.4) for  $S_L$  but with one difference: the scanning process does not start from  $G_R[1]$  but from  $S_L[1]$  again. We proceed as follow.

1. We scan the first  $n/c$  pointers of  $S_L$  (they correspond to subarray  $\text{Tab}'_L$ , which has not yet been used to encode the  $\alpha$ - $\beta$  table). The scanning process is the same one of the fourth phase, except that we use  $D_R$  as directory and  $\text{Tab}_L''$  for the  $\alpha$ - $\beta$  table.
2. After the first  $n/c$  pointers of  $S_L$  have been scanned, we move the  $\alpha$ - $\beta$  table encoding from  $\text{Tab}_L''$  to  $\text{Tab}'_L$  and recover the lexicographical order of the pointers in  $\text{Tab}_L''$ . Then, we continue the scanning of the rest of  $S_L$ .
3. Finally, the scanning process arrives to  $G_R$ . After  $G_R$  is sorted we merge  $G_R$  and  $D_R$  as we merged  $G_L$  and  $D_L$  in the fourth phase and we are done.

Let us point out one peculiar aspect of this last process. Since during the scan we use  $D_R$  as directory, the suffixes whose pointers reside in  $S_L$  *will not be moved again*. That is because  $D_R$  has been built using pointers to  $\beta$ -suffixes whose first characters are always different from the ones of suffixes with pointers in  $S_L$ . For this reason, during the second scan of  $S_L$ , any search in  $D_R$  for a suffix whose pointer is in  $S_L$  *will always fail* and nothing will be done to the pointer for that suffix (correctly, since  $S_L$  is already in order).

To summarize, we get Theorem [□](#).

## 4 Concluding Remarks

We have presented first known inplace algorithm for suffix sorting, i.e., an algorithm that uses  $O(1)$  workspace beyond what is needed for the input  $T$  and output  $S$ . This algorithm is optimal for the general alphabet. Ultimately we would like to see simpler algorithms for this problem. Also, an interesting case is one in which the string elements are drawn from an integer alphabet. Then we can assume that each  $T[i]$  is stored in  $\lceil \log n \rceil$  bits and  $O(1)$  time bit operations

are allowed on such elements. In that case, known suffix tree algorithms solve suffix sorting in  $O(n)$  time and use  $O(n)$  workspace in addition to  $T$  and  $S$  [11]. We leave it open to design inplace algorithms for this case in  $o(n \log n)$  time and ultimately, in even  $O(n)$  time.

## References

1. Farach, M.: Optimal Suffix Tree Construction with Large Alphabets. In: FOCS 1997, pp. 137–143 (1997)
2. Ferragina, P., Manzini, G.: Engineering a lightweight suffix array construction algorithm. In: Proc. ESA (2002)
3. Gusfield, D.: Algorithms on strings, trees and sequences: Computer Science and Computational Biology. Cambridge Univ Press, Cambridge (1997)
4. Kärkkäinen, J., Sanders, P.: Simple linear work suffix array construction. Int. Colloquium on Automata, Languages and Programming 2719, 943–955 (2003)
5. Kärkkäinen, J., Sanders, P., Burkhardt, S.: Linear work suffix array construction. Journal of the ACM (in press)
6. Ko, P., Aluru, S.: Space efficient linear time construction of suffix arrays. In: Baeza-Yates, R.A., Chávez, E., Crochemore, M. (eds.) CPM 2003. LNCS, vol. 2676, pp. 200–210. Springer, Heidelberg (2003)
7. Ian Munro, J.: An implicit data structure supporting insertion, deletion, and search in  $O(\log^2 n)$  time. Journal of Computer and System Sciences 33(1), 66–74 (1986)
8. Salowe, J., Steiger, W.: Simplified stable merging tasks. Journal of Algorithms 8(4), 557–571 (1987)

# Maximal Infinite-Valued Constraint Languages

Manuel Bodirsky<sup>1</sup>, Hubie Chen<sup>2</sup>, Jan Kára<sup>3</sup>, and Timo von Oertzen<sup>4</sup>

<sup>1</sup> Department of Algorithms and Complexity, Humboldt University, Berlin  
bodirsky@informatik.hu-berlin.de

<sup>2</sup> Department of Applied Mathematics, Charles University, Prague  
kara@kam.mff.cuni.cz

<sup>3</sup> Departament de Tecnologies de la Informació i les Comunicacions  
hubie.chen@upf.edu

<sup>4</sup> Max-Planck-Institute for Human Development, Berlin  
vonoertzen@mpib-berlin.mpg.de

**Abstract.** We systematically investigate the computational complexity of constraint satisfaction problems for constraint languages over an infinite domain. In particular, we study a generalization of the well-established notion of *maximal constraint languages* from finite to infinite domains. If the constraint language can be defined with an  $\omega$ -categorical structure, then maximal constraint languages are in one-to-one correspondence to *minimal oligomorphic clones*. Based on this correspondence, we derive general tractability and hardness criteria for the corresponding constraint satisfaction problems.

## 1 Introduction

One of the research goals in constraint satisfaction is to determine the constraint languages whose constraint satisfaction problem (CSP) can be solved by a polynomial time algorithm (we will call such languages *tractable*), and the languages for which the constraint satisfaction problem is NP-hard. In the last decade, a lot of progress was made in this direction, in particular if the domain  $D$  of the constraint language is finite. Particularly stimulating has been the observation that the computational complexity of a constraint satisfaction problem is for finite  $D$  determined by a set of closure operations, which forms an object known as *polymorphism clone* in universal algebra. The line of research that uses this connection to universal algebra is also known as the *algebraic approach* to constraint satisfaction; see e.g. [10] for a recent account.

The complexity classification of CSPs for finite domains  $D$  is still not complete. However, based on the algebraic approach, the complexity classification for *maximal constraint languages* has been completed recently [8, 9]. Roughly speaking, a constraint language is maximal if it is as large as possible without expressing *all* relations on  $D$ . The notion of a maximal constraint language was previously only used for finite domains  $D$ .

Compared to constraint satisfaction with finite domains, there are not as many systematic results for infinite-valued CSPs. One of the outstanding exceptions is

the complexity classification for the tractable sub-languages of Allen's interval algebra. Allen's interval algebra is a (binary) constraint language that allows to specify relative positions of intervals over the rational numbers, and is in its unrestricted form NP-complete [1]. However, there are many tractable sub-languages, which were determined in a series of papers, most notably in [19,20].

Many constraint languages in temporal and spatial reasoning, but also constraint languages studied in computational linguistics and computational biology are  $\omega$ -categorical. The CSPs for the fragments of Allen's interval algebra, for instance, can all be formulated with  $\omega$ -categorical constraint languages. The concept of  $\omega$ -categoricity is of central importance in model theory. It turns out that the algebraic approach to constraint satisfaction can be applied not only to constraint languages over a finite domain, but also to  $\omega$ -categorical constraint languages [2,4,6]. From the model-theoretic point of view,  $\omega$ -categoricity is a severe restriction. However, the class of CSPs that can be formulated with  $\omega$ -categorical structures is very rich. It contains for instance all CSPs (for a constraint language over an arbitrary infinite domain) in MMSNP [4], a fragment of existential second order logic, which was introduced in the context of constraint satisfaction in [13]. We will later also see how  $\omega$ -categorical structures can be used to model problems about solving equations over infinite-dimensional vector spaces.

*Contributions.* In this paper, we introduce and investigate a notion of maximal constraint languages that extends to infinite domains  $D$ . For finite  $D$ , our definition of maximal constraint languages essentially coincides with the well-established notion of maximal constraint languages in [8,9]. We will use the fact that maximal  $\omega$ -categorical constraint languages are in one-to-one correspondence to minimal oligomorphic clones, and prove that the tractable maximal constraint languages are of three types: either they have a polymorphism that is a *unary constant operation*, a *quasi majority operation*, or an *essentially binary oligopotent operation*. In the first case, the constraint satisfaction problem is trivial and tractable. In the second case, tractability follows from a result in [4]. Therefore, all maximal constraint languages of unknown computational complexity are preserved by an essentially binary operation.

Another main contribution is a strong tractability criterion which shows that  $\omega$ -categorical constraint languages with a certain binary polymorphism can be solved in polynomial time. This class also contains many maximal constraint languages. As demonstrated in Section 8, our condition also captures and extends tractability results in qualitative temporal and spatial reasoning, and it provides an universal-algebraic perspective on these results.

## 2 The Constraint Satisfaction Problem

We first recall fundamental concepts and notation used throughout the text; the book of Hodges [16] might serve as an introduction. A *relational language*  $\tau$  is a (here always at most countable) set of *relation symbols*  $R_i$ , each associated with an *arity*  $k_i$ . A (*relational*) *structure*  $\Gamma$  over the relational language  $\tau$  (also called  $\tau$ -*structure*) is a set  $D_\Gamma$  (the *domain*) together with a relation  $R_i \subseteq D_\Gamma^{k_i}$

for each relation symbol of arity  $k_i$ . If necessary, we write  $R^\Gamma$  to indicate that we are talking about the relation  $R$  belonging to the structure  $\Gamma$ . For simplicity, we denote both a relation symbol and its corresponding relation with the same symbol. For a  $\tau$ -structure  $\Gamma$  and  $R \in \tau$  it is convenient to say that  $R(u_1, \dots, u_k)$  holds in  $\Gamma$  iff  $(u_1, \dots, u_k) \in R$ . We sometimes use the shortened notation  $\bar{x}$  for a vector  $x_1, \dots, x_n$  of any length. Sometimes we do not distinguish between the symbol for a relational structure  $\Gamma$  and its domain  $D_\Gamma$ . If we add relations to a given structure  $\Gamma$  we call the resulting structure  $\Gamma'$  an *expansion* of  $\Gamma$ , and we call  $\Gamma$  a *reduct* of  $\Gamma'$ .

Let  $\Gamma$  and  $\Gamma'$  be  $\tau$ -structures. A *homomorphism* from  $\Gamma$  to  $\Gamma'$  is a function  $f$  from  $D_\Gamma$  to  $D_{\Gamma'}$  such that for each  $n$ -ary relation symbol in  $\tau$  and each  $n$ -tuple  $\bar{a}$ , if  $\bar{a} \in R^\Gamma$ , then  $(f(a_1), \dots, f(a_n)) \in R^{\Gamma'}$ . In this case we say that the mapping  $f$  *preserves* the relation  $R$ . If there is a homomorphism from  $\Gamma$  to  $\Gamma'$  and a homomorphism from  $\Gamma'$  to  $\Gamma$ , we say that  $\Gamma$  and  $\Gamma'$  are *homomorphically equivalent*. Homomorphisms from  $\Gamma$  to  $\Gamma$  are called *endomorphisms*. A homomorphism is called an *embedding*, if it is injective and satisfies the stronger condition that for each  $n$ -ary relation symbol in  $\tau$  and each  $n$ -tuple  $\bar{a}$ ,  $\bar{a} \in R^\Gamma$  if and only if  $(f(a_1), \dots, f(a_n)) \in R^{\Gamma'}$ . An *isomorphism* is a surjective embedding. Isomorphisms from  $\Gamma$  to  $\Gamma$  are called *automorphisms*. The set of all automorphisms  $\text{Aut}(\Gamma)$  of a structure  $\Gamma$  is a group with respect to composition.

*The constraint satisfaction problem.* Let  $\Gamma$  be a structure with the relational language  $\tau$ . The constraint satisfaction problem (CSP) for the *template*  $\Gamma$  is the following computational problem:

### CSP( $\Gamma$ )

INSTANCE: A finite structure  $S$  of the same relational language  $\tau$  as the template  $\Gamma$ .

QUESTION: Is there a homomorphism from  $S$  to  $\Gamma$ ?

The elements of the finite input structure  $S$  are also called the *variables* of the CSP. In order to study the computational complexity of this problem, we have to encode the input structure  $S$  as a finite string over the alphabet  $\{0, 1\}$ . However, if we assume that the language  $\tau$  is finite, the exact choice of the representation does not matter (since the relational language is fixed and in particular does not grow with the size of the input). For infinite constraint languages  $\tau$ , we say that  $\text{CSP}(\Gamma)$  is tractable if and only if  $\text{CSP}(\Gamma')$  is tractable for all all reducts  $\Gamma'$  of  $\Gamma$  having a *finite* relational language. This definition is commonly used also for constraint satisfaction over finite domains [10].

To study the computational complexity of the CSP, reductions from one constraint language to another can be described conveniently using the notion of *primitive positive definability* from logic. A formula is called *primitive positive (pp)*, if it has the form  $\exists x_1 \dots x_k. \psi_1 \wedge \dots \wedge \psi_l$ , where  $\psi_i$  is atomic (it might be of the form  $x = y$ , i.e., we always include equality in our language). The atomic formulas might contain free variables and existentially quantified variables from

$x_1, \dots, x_k$ . As usual, every formula with  $k$  free variables defines on a structure  $\Gamma$  a  $k$ -ary relation. Primitive positive definability of relations is an important concept in constraint satisfaction, because pp-definable relations can be 'simulated' by the constraint satisfaction problem. The following is frequently used in hardness proofs for constraint satisfaction problems; see e.g. [10].

**Lemma 1.** *Let  $\Gamma$  be a relational structure, and let  $R$  be a relation that has a primitive positive definition in  $\Gamma$ . Then the constraint satisfaction problems of  $\Gamma$  and of the expansion of  $\Gamma$  by  $R$  have the same computational complexity up to logspace reductions.*

The (universal-) algebraic approach to constraint satisfaction relies on the fact that pp-definability can be characterized by preservation under so-called *polymorphisms*; we introduce these concepts in Section 4.

### 3 Preliminaries from Model Theory

We first recall fundamental concepts from model theory, which are standard, see e.g. [16]. A relational structure over a countably infinite domain is called  $\omega$ -categorical if the first-order theory of  $\Gamma$  has only one countable model up to isomorphism. The following deep theorem discovered independently by Engeler, Ryll-Nardzewski, and Svenonius (see [16]) describes these structures in algebraic terms.

An *orbit* of a  $k$ -tuple  $t$  in  $\Gamma$  is the set of all tuples of the form  $(\pi(t_1), \dots, \pi(t_k))$ , where  $\pi$  is an automorphism of  $\Gamma$ . The automorphism group  $G$  of  $\Gamma$  is called *oligomorphic*, if for each  $k \geq 1$ , there are finitely many orbits of  $k$ -tuples in  $G$ .

**Theorem 1 (Engeler, Ryll-Nardzewski, Svenonius; see e.g. [16]).** *A countable relational structure is  $\omega$ -categorical if and only if the automorphism group of  $\Gamma$  is oligomorphic. A relation  $R$  has a first-order definition in an  $\omega$ -categorical structure  $\Gamma$  if and only if  $R$  is preserved by all automorphisms of  $\Gamma$ .*

An  $\omega$ -categorical structure  $\Gamma$  is called *model-complete* if every embedding from  $\Gamma$  into  $\Gamma$  preserves all first-order formulas. It is called *homogeneous* (in the literature sometimes also *ultra-homogeneous*), if all isomorphisms between finite induced substructures of  $\Gamma$  can be extended to automorphisms of  $\Gamma$ . It is well-known [16] that an  $\omega$ -categorical structure is homogeneous if and only if it has *quantifier-elimination*, i.e., every first-order formula is over  $\Gamma$  equivalent to a quantifier-free formula. Homogeneous  $\omega$ -categorical structures are always model-complete [16]. We say that an  $\omega$ -categorical structure  $\Gamma$  is a *core*, if every endomorphism of  $\Gamma$  is an embedding. We say that  $\Delta$  is a *core* of  $\Gamma$  if  $\Delta$  is a core and homomorphically equivalent to  $\Gamma$ .

**Theorem 2 (from [2]).** *Every  $\omega$ -categorical structure  $\Gamma$  has a model-complete core  $\Delta$ , which is unique up to isomorphism, and which is either finite or  $\omega$ -categorical. Every relation consisting of a single orbit of  $k$ -tuples of  $\Delta$  has a primitive positive definition in  $\Delta$ .*

Since a model-complete core  $\Delta$  of  $\Gamma$  is unique up to isomorphisms, we call  $\Delta$  *the core* of  $\Gamma$ . Clearly,  $\Gamma$  and  $\Delta$  have the same constraint satisfaction problem, and we can therefore always assume that templates of constraint satisfaction problems are model-complete cores. One of the reasons why it is convenient to assume that  $\Gamma$  is a model-complete core is the following.

**Lemma 2 (from [2]).** *Let  $\Gamma$  be a model-complete  $\omega$ -categorical core, and let  $\Gamma'$  be the expansion of  $\Gamma$  by a unary singleton relation  $C = \{c\}$ . If  $\text{CSP}(\Gamma)$  is tractable, then so is  $\text{CSP}(\Gamma')$ .*

## 4 Preliminaries from Universal Algebra

To explore the expressive power of a constraint language, we make use of universal algebraic techniques. We give a very short but self-contained introduction to clones on infinite domains.

Let  $D$  be an infinite set, and let  $O^{(k)}$  be a subset of operations from  $D^k$  to  $D$ , for  $k \geq 1$ . The symbol  $O$  denotes  $\bigcup_{k=1}^{\infty} O^{(k)}$ . An operation  $\pi \in O^{(k)}$  is called a *projection* if for some fixed  $i \in \{1, \dots, k\}$  and for all  $k$ -tuples  $x$  we have the identity  $\pi(x_1, \dots, x_k) = x_i$ . The *composition* of a  $k$ -ary operation  $f$  and  $k$  operations  $g_1, \dots, g_k$  of arity  $n$  is an  $n$ -ary operation defined by

$$f(g_1, \dots, g_k)(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n)).$$

We say that an operation  $f \in O^{(k)}$  is *interpolated* by a set  $F \subseteq O$  if for every finite subset  $B$  of  $D$  there is some operation  $g \in F$  such that  $f(t) = g(t)$  for every  $t \in B^k$ . The set of all operations that are interpolated by  $F$  is denoted by  $I(F)$ .

A subset  $F$  of  $O$  is called a *clone* if it contains all *projections* and is *closed under composition*. It is called a *local clone* if  $I(F) = F$ . The smallest clone that contains  $F$  is called the clone *generated* by  $F$ , and denoted by  $G(F)$ . The smallest *local clone* that contains  $F$  is called the clone *locally generated* by  $F$ , and denoted by  $L(F)$ . The following seems to be folklore, and is easy to see. (Note that  $L(F) = G(I(F))$  is in general not true.)

**Proposition 1.** *For all  $F \subseteq O$  we have that  $L(F) = I(G(F))$ .*

The connection to the expressive power of constraint languages is as follows. The (*direct-, categorical-, or cross-*) *product*  $\Gamma_1 \times \Gamma_2$  of two relational  $\tau$ -structures  $\Gamma_1$  and  $\Gamma_2$  is a  $\tau$ -structure on the domain  $D_{\Gamma_1} \times D_{\Gamma_2}$ . For all relations  $R \in \tau$  the relation  $R((x_1, y_2), \dots, (x_k, y_k))$  holds in  $\Gamma_1 \times \Gamma_2$  iff  $R(x_1, \dots, x_k)$  holds in  $\Gamma_1$  and  $R(y_1, \dots, y_k)$  holds in  $\Gamma_2$ . Homomorphisms from  $\Gamma^k = \Gamma \times \dots \times \Gamma$  to  $\Gamma$  are called *polymorphisms* of  $\Gamma$ . It is well-known and easy to prove that the set of all polymorphisms of a relational structure  $\Gamma$  with domain  $D$ , denoted by  $\text{Pol}(\Gamma)$ , is a local clone (with domain  $D$ ).

The algebraic approach is based on the following observation, which shows together with Lemma 1 that the computational complexity of a constraint

satisfaction problem with template  $\Gamma$  is determined by the polymorphisms of  $\Gamma$ . If  $F$  is a clone, we denote by  $\text{Inv}(F)$  the set of relations that are preserved by  $F$ .

**Theorem 3 (from [6]).** *Let  $\Gamma$  be an  $\omega$ -categorical structure. Then a relation is primitive positive definable in  $\Gamma$  if and only if it is preserved by all polymorphisms of  $\Gamma$ . In other words,  $\text{Inv}(\text{Pol}(\Gamma))$  is the set of all primitive positive definable relations of  $\Gamma$ .*

This motivates the study of polymorphism clones of  $\omega$ -categorical structures. A clone  $F$  on a countable set  $D$  is called *oligomorphic*, if the permutations of  $D$  that are contained in  $F$  form an oligomorphic permutation group. Theorem 1 asserts that the polymorphism clones of  $\omega$ -categorical structures are oligomorphic. Conversely, a locally closed oligomorphic clone is the polymorphism clone of an  $\omega$ -categorical structure; see [3].

An operation of an oligomorphic clone  $F$  is called *elementary* if it is locally generated by the permutations in  $F$ . Clearly, for finite clones, the elementary operations are the operations that are composed of a projection with a permutation. Note that all endomorphisms of a model-complete  $\omega$ -categorical core are elementary.

**Proposition 2 (from [3]).** *If all polymorphisms of an  $\omega$ -categorical structure  $\Gamma$  are locally generated by the automorphisms of  $\Gamma$ , then every first-order formula is in  $\Gamma$  equivalent to a primitive positive formula.*

We now define several other important properties of  $k$ -ary operations. A  $k$ -ary operation  $f$  is

- *idempotent* iff  $f(x, \dots, x) = x$ ;
- *oligopotent* iff  $g(x) := f(x, \dots, x)$  is elementary
- *essentially unary* iff there is a unary operation  $f_0$  such that  $f(x_1, \dots, x_k) = f_0(x_i)$  for some  $i \in \{1, \dots, k\}$ ;
- *essential* iff  $f$  is not essentially unary;
- a *quasi near-unanimity operation* (short, *qnu-operation*) iff  $f(x, \dots, x) = f(x, \dots, x, y) = \dots = f(x, \dots, x, y, x, \dots, x) = \dots = f(y, x, \dots, x)$ ;
- a *quasi majority operation* iff  $f$  is a ternary quasi near-unanimity operation;
- a *quasi semiprojection* iff there is an essentially unary operation  $g$  such that  $f(x_1, \dots, x_k) = g(x_1, \dots, x_k)$  whenever  $|\{x_1, \dots, x_k\}| < k$ .

An idempotent quasi majority, quasi near-unanimity, and quasi semiprojection is known as majority, near-unanimity operation, and semiprojection, respectively. If all operations of a clone are elementary, essentially unary, or oligopotent, then we say that the clone is elementary, essentially unary, or oligopotent, respectively.

*Minimal Clones.* Important questions in universal algebra and useful tools for constraint satisfaction arise from the notion of *minimal clones*. A (proper) subclone  $F'$  of a clone  $F$  is a clone on the same domain as  $F$ , and the operations of  $F'$  form a (proper) subset of the operations of  $F$ . An oligomorphic clone  $F$  is called *minimal*, if in every proper oligomorphic subclone every operation is elementary.



The following is the oligomorphic analog of a result proved by Rosenberg [22] for clones with a finite domain<sup>1</sup>.

**Theorem 4 (of [3]).** *Every minimal oligomorphic clone  $F$  is locally generated by the permutations from  $F$  and a non-elementary operation that is of one of the following types*

1. a unary operation  $f$  such that  $f(f)$  and the permutations in  $F$  locally generate  $f$ ;
2. a binary oligopotent operation;
3. a ternary oligopotent quasi majority operation;
4. a  $k$ -ary oligopotent quasi semi-projection, for  $k > 2$ .

We also need the following.

**Theorem 5 (of [3]).** *Let  $\Gamma$  be an  $\omega$ -categorical model-complete core. If the polymorphism clone  $F$  of  $\Gamma$  contains a non-elementary operation, then  $F$  also contains a minimal oligomorphic clone.*

## 5 Hardness Criteria for CSPs

We show that if a constraint language is not preserved by polymorphisms of a special kind, the corresponding constraint satisfaction problem must be NP-hard.

**Theorem 6.** *Let  $\Gamma$  be an  $\omega$ -categorical structure. Then either  $\Gamma$  has a finite core, or  $\text{CSP}(\Gamma)$  is NP-hard, or  $\Gamma$  has a polymorphism  $f$  of one of the following types.*

- an oligopotent binary operation
- an oligopotent ternary quasi majority operation
- a  $k$ -ary oligopotent quasi semi-projection, for  $k \geq 3$

It was shown in [17] that for constraint languages over finite domains the semiprojections alone do not guarantee tractability. The same holds even for  $\omega$ -categorical constraint languages and quasi semiprojections.

**Proposition 3.** *Let  $\Gamma$  be an  $\omega$ -categorical core. If all polymorphisms of  $\Gamma$  are locally generated by quasi semiprojections, then  $\text{CSP}(\Gamma)$  is NP-hard.*

## 6 Maximal Constraint Languages

A constraint language  $\Gamma$  is called *complete*, if every first-order definable relation in  $\Gamma$  also has a primitive positive definition in  $\Gamma$ . We call an incomplete constraint language *maximal*, if adding any relation to the language that was

<sup>1</sup> Note that one of the five cases presented by Rosenberg can not occur for minimal oligomorphic clones, essentially because  $\omega$ -categorical model-complete cores can not have a ternary polymorphism that satisfies the identities  $f(y, x, x) = f(x, x, y) = f(y, y, y)$ ; see [3].

not primitive positive definable before turns the constraint language into a complete constraint language. We will see that for  $\omega$ -categorical structures, maximal constraint languages precisely correspond to minimal locally closed clones.

This definition of maximality coincides with the well-established notion of maximality of constraint languages for finite templates (see [9,10]), if we assume that the constraint language additionally contains for every element  $a \in D$  a symbol  $R_a$  that denotes the unary relation  $\{a\}$ . Every CSP with a finite domain is polynomial-time equivalent to the CSP where the language has been expanded by all singleton relations as described above [10].

We briefly recall the definition of maximality given in [9]. For finite domains  $D$  a constraint language is called *complete* if every relation over  $D$  has a primitive positive definition over the constraint language. As before, an incomplete constraint language is called *maximal* if adding any relation to the language that was not primitive positive definable before turns the constraint language into a complete constraint language. Once we have singleton relations for all elements in our constraint language, this definition of completeness coincides with the definition shown above, because in such constraint languages every relation has a first-order definition, which is easy to see. Therefore, our definition and the standard definition of maximality essentially coincide on finite templates.

**Proposition 4 (of [3]).** *Let  $\Gamma$  be an  $\omega$ -categorical constraint language that is not complete, and let  $F$  be the polymorphism clone of  $\Gamma$ . Then the following are equivalent:*

1. *The language of  $\Gamma$  is maximal;*
2.  *$F$  is minimal, i.e., every proper oligomorphic subclone of  $F$  is elementary;*
3. *Every non-elementary operation in  $F$  together with the permutations in  $F$  locally generates  $F$ .*

Theorem 6 now specializes to the following.

**Theorem 7.** *Let  $\Gamma$  be an  $\omega$ -categorical structure with a maximal language. Then one of the following cases applies.*

1.  *$CSP(\Gamma)$  is NP-hard;*
2.  *$\Gamma$  has a constant polymorphism, and  $CSP(\Gamma)$  is tractable;*
3. *The polymorphisms of  $\Gamma$  are locally generated by the automorphisms of  $\Gamma$  and a non-elementary binary operation, and  $\Gamma$  is a core;*
4. *The polymorphism clone of  $\Gamma$  contains a quasi majority operation, and  $CSP(\Gamma)$  is tractable.*

Note that all the maximal  $\omega$ -categorical constraint languages of unknown computational complexity are from the third case of Theorem 7.

## 7 Tractability

In this section we present a new general tractability criterion based on preservation under a binary polymorphism.

If  $\Gamma = (D; R_1, \dots)$  is a relational structure, we denote by  $\Gamma^c$  the expansion of  $\Gamma$  that also contains the complement for each relation in  $\Gamma$ . We say that a relation  $R$  has a *quantifier-free Horn definition* in  $\Gamma$  if  $R$  can be defined by a quantifier-free first-order formula over the signature of  $\Gamma$  that is in conjunctive normal form in which every clause contains at most one positive literal.

**Theorem 8.** *Let  $\Gamma$  be an  $\omega$ -categorical homogeneous structure. If there is an isomorphism  $i$  between  $\Gamma^2$  and  $\Gamma$ , then*

- *the binary operation  $i : D^2 \rightarrow D$  and the automorphisms of  $\Gamma$  locally generate a minimal oligomorphic clone  $F$ ; hence,  $\Delta = \text{Inv}(F)$  is a maximal constraint language;*
- *all relations in  $\Delta$  have a quantifier-free Horn definition in  $\Gamma$ ;*
- *if  $\text{CSP}(\Gamma^c)$  is tractable, then  $\text{CSP}(\Delta)$  is tractable as well.*

Applications of Theorem 8 for concrete constraint languages and computational problems can be found in Section 8. We would like to remark that our tractability result is a new application of the universal-algebraic approach to constraint satisfaction, and indeed we have linked tractability of constraint languages to the existence of a binary polymorphism that locally generates a minimal oligomorphic clone. Note that this tractability result is specific to infinite domains, because for constraint languages  $\Gamma$  over a finite domain  $\Gamma^2$  cannot possibly be isomorphic to  $\Gamma$ . For infinite structures  $\Gamma$  the situation that  $\Gamma$  is isomorphic to  $\Gamma^2$  is not so rare; it is e.g. well-known that the set of models of a universal first-order Horn theory is preserved under direct products [16].

## 8 Applications

### 8.1 Solving Equations over Infinite Vector Spaces

Solving equations with equalities and disequalities for an infinite-dimensional vector space over a finite field can be formulated as a constraint satisfaction problem with an  $\omega$ -categorical template. The tractability result we present here is new.

Let  $F_q$  be a finite field with elements  $s_0, \dots, s_{q-1}$  and let  $V$  be a countably infinite vector space over  $F_q$ . Then  $V$  is unique up to isomorphism [16]. We consider the following relational structure  $\Gamma_V := (V, R^+, R^{s_0}, \dots, R^{s_{q-1}})$ , where the relations are defined as follows.

$$R^+(x, y, z) \equiv (x + y = z)$$

$$R^{s_i}(x, y) \equiv (y = s_i x)$$

The structure  $\Gamma_V$  is homogeneous and  $\omega$ -categorical [16]. Hence, its automorphism group is oligomorphic; in fact, the automorphism group is one of the groups known as the *classical infinite groups*, which have many remarkable properties [14]. Clearly, the constraint satisfaction problem for  $\Gamma_V$  is trivial, because it has a constant endomorphism. But in the expansion  $\Gamma_V^c$  of  $\Gamma_V$  we can find

primitive positive definitions of the relation defined by  $x \neq y$ , and it is not difficult to show that  $\Gamma_V^c$  is a core. It is also not difficult to come up with an algorithm for  $\text{CSP}(\Gamma_V^c)$ , essentially by Gaussian elimination.

It is well-known that  $V^2$ , the direct product of the algebraic object  $V$  with itself, is isomorphic to  $V$ ; similarly, we have that  $\Gamma_V^2$  (as defined in Section 4) is isomorphic to  $\Gamma_V$ . Let  $i_V$  be the isomorphism, and let  $C_V$  be the oligomorphic clone that is locally generated by  $i_V$  and the automorphisms of  $\Gamma_V$ . Then Theorem 8 implies that  $\text{Inv}(C_V)$  is a maximal constraint language. Since  $\text{CSP}(\Gamma_V^c)$  is tractable as well, we can apply Theorem 8 and find that  $\text{CSP}(\text{Inv}(C_V))$  is tractable. This is a more interesting result, saying that we can find an efficient algorithm for Horn clauses of equations for infinite vector spaces over a finite field.

### 8.2 CSPs for the Universal Triangle-Free Graph

Let  $\mathcal{A}$  be a countably infinite graph  $(D, E)$  that satisfies the following property:  $\mathcal{A}$  does not contain a triangle (i.e., does not contain  $K_3$  as a subgraph), and for all finite disjoint sets  $S, I \subset D$  where  $I$  forms an independent set in  $\mathcal{A}$  we find a vertex  $v \notin S \cup I$  such that  $v$  is connected to all vertices in  $I$  and to no vertex in  $S$ .

It is well known that such a graph must be unique up to isomorphism, and is homogeneous and  $\omega$ -categorical; it is known as the *universal homogeneous triangle-free graph*. Clearly,  $\text{CSP}(\mathcal{A}^c)$  can be solved in polynomial time: we simply have to check that the input instance does not contain two literals  $E(x, y)$  and  $\neg E(x, y)$  at the same time, and that  $E$  is without loops and triangle-free. One can easily verify that  $\mathcal{A}^2$  is isomorphic to  $\mathcal{A}$ ; let  $i$  be the isomorphism. By Theorem 8, we have found another maximal  $\omega$ -categorical constraint language, and the corresponding CSP can be solved in polynomial time.

### 8.3 Spatial Reasoning

One of the most fundamental spatial reasoning formalisms is the RCC-5 calculus (also known as the containment algebra in the theory of relation algebras [11]). The maximally tractable sublanguages of the binary constraint language for RCC-5 have been determined in [18, 21]. Let  $\mathbb{B}_0$  be the countable atomless boolean ring without an identity elements. This structure is unique up to isomorphism and  $\omega$ -categorical, see for example [12]. It is straightforward to verify that  $(\mathbb{B}_0)^2$  is isomorphic to  $\mathbb{B}_0$ . We are interpreting the elements of this boolean ring as non-empty sets (regions), where  $A + B$  denotes the symmetric difference, and  $A \cdot B$  the intersection of  $A$  and  $B$ . Now, consider the relational structure  $\Sigma$  over the domain of  $\mathbb{B}_0$  with the two binary relations called DR and PP (these are the traditional names in RCC-5). With the interpretation of the elements of  $\mathbb{B}_0$  being sets, they are defined as follows: DR( $X, Y$ ) iff  $X \cap Y = \emptyset$  and PP( $X, Y$ ) iff  $X \subset Y$ .

The constraint satisfaction problem for  $\text{CSP}(\Sigma^c)$  is in P [21].  $\Sigma^2$  is isomorphic to  $\Sigma$  as well, and it again follows by Theorem 8 that the constraint language whose relations have a Horn definition in  $\Sigma$  is tractable. We would like to remark that Nebel and Renz [21] determined a largest tractable fragment of RCC-5, and

that it follows from the result in [18] that this fragment is the unique largest tractable fragment that contains the basic relations DR and PP. But note that RCC-5 only contains binary relations, whereas our maximal language contains relations of arbitrary arity.

## 8.4 Temporal Reasoning

We present another maximal constraint language from the field of temporal reasoning. It will serve us as an example of a maximal constraint language whose polymorphism clone is locally generated by a binary injective operation, but which has an NP-complete CSP.

One of the basic structures for temporal reasoning is  $(\mathbb{Q}, <)$ , the set of rational numbers, ordered by  $<$ . This structure is an unbounded and dense linear order on countably many vertices, and it is uniquely described by these properties, up to isomorphism. Note that  $(\mathbb{Q}, <)^2$  is clearly *not* isomorphic to  $(\mathbb{Q}, <)$ .

Consider a binary operation  $\text{lex}$  on  $\mathbb{Q}$  satisfying  $\text{lex}(a, b) < \text{lex}(a', b')$  if either  $a < a'$ , or  $a = a'$  and  $b < b'$ . Note that every operation  $\text{lex}$  satisfying these conditions is by definition injective. Let  $F$  be the clone generated by  $\text{lex}$  and the automorphisms of  $(\mathbb{Q}, <)$ . The problem  $\text{CSP}(\text{Inv}(F))$  is NP-complete. This is because all operations in  $F$  preserve the Betweenness relation defined by the formula  $(x < y < z) \vee (z < y < x)$ , and the CSP for the Betweenness relation is a well-known NP-complete problem [15]. It is not hard to show that  $F$  is a minimal oligomorphic clone.

For temporal reasoning with *partially ordered time* as studied by Broxvall and Jonsson [7] there are four maximally tractable classes, called  $\mathcal{S}_A, \mathcal{S}_B, \mathcal{S}_C, \mathcal{S}_D$  of so-called *disjunctive constraint languages*; for terminology in this paragraph, we refer to [7]. These constraint languages all have a first-order definition in the so-called countable universal homogeneous partial order, denoted here by  $P$ ; see [23]. It is easy to verify that there is an isomorphism  $i_P$  between  $P^2$  and  $P$ . The language  $\mathcal{S}_D$  has a constant endomorphism, and  $\mathcal{S}_B$  has a core with a first-order definition in  $(\mathbb{Q}, <)$ . It can be verified that the language  $\mathcal{S}_C$  is preserved by  $i_P$ ; in fact, the relations with a first-order definition in  $P$  that are preserved by  $i_P$  strictly contain the relations from  $\mathcal{S}_C$ . Since  $P^c$  is tractable (this is easy to see) tractability and maximality of again follows from Theorem [8].

## References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11), 832–843 (1983)
2. Bodirsky, M.: Cores of countably categorical structures. *Logical Methods in Computer Science (LMCS)* (2007), doi:DOI: 10.2168/LMCS-3(1:2)
3. Bodirsky, M., Chen, H.: Oligomorphic clones. *Algebra Universalis* (to appear, 2007)
4. Bodirsky, M., Dalmau, V.: Datalog and constraint satisfaction with infinite templates. In: Durand, B., Thomas, W. (eds.) *STACS 2006*. LNCS, vol. 3884, pp. 646–659. Springer, Heidelberg (2006)

5. Bodirsky, M., Kára, J.: The complexity of equality constraint languages. In: Grigoriev, D., Harrison, J., Hirsch, E.A. (eds.) CSR 2006. LNCS, vol. 3967, Springer, Heidelberg (2006)
6. Bodirsky, M., Nešetřil, J.: Constraint satisfaction with countable homogeneous templates. *Journal of Logic and Computation* 16(3), 359–373 (2006)
7. Broxvall, M., Jonsson, P.: Point algebras for temporal reasoning: Algorithms and complexity. *Artif. Intell.* 149(2), 179–220 (2003)
8. Bulatov, A.: A graph of a relational structure and constraint satisfaction problems. In: *Proceedings of the 19th IEEE Annual Symposium on Logic in Computer Science (LICS'04)*, Turku, Finland (2004)
9. Bulatov, A., Krokhin, A., Jeavons, P.: The complexity of maximal constraint languages. In: *Proceedings of STOC '01*, pp. 667–674 (2001)
10. Bulatov, A., Krokhin, A., Jeavons, P.G.: Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing* 34, 720–742 (2005)
11. Düntsch, I.: algebras and their application in temporal and spatial reasoning. *Artificial Intelligence Review* 23, 315–357 (2005)
12. Evans, D.: Examples of  $\aleph_0$ -categorical structures. In: *Automorphisms of first-order structures*, pp. 33–72 (1994)
13. Feder, T., Vardi, M.: The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing* 28, 57–104 (1999)
14. Gardener, T.: Infinite dimensional classical groups. *J. London Math. Soc* 51, 219–229 (1995)
15. Garey, Johnson: *A Guide to NP-completeness*. CSLI Press, Stanford (1978)
16. Hodges, W.: *A shorter model theory*. Cambridge University Press, Cambridge (1997)
17. Jeavons, P., Cohen, D., Gyssens, M.: Closure properties of constraints. *Journal of the ACM* 44(4), 527–548 (1997)
18. Jonsson, P., Drakengren, T.: A complete classification of tractability in RCC-5. *J. Artif. Intell. Res.* 6, 211–221 (1997)
19. Krokhin, A.A., Jeavons, P., Jonsson, P.: Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *Journal of the ACM* 50(5), 591–640 (2003)
20. Nebel, B., Bürckert, H.-J.: Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM* 42(1), 43–66 (1995)
21. Renz, J., Nebel, B.: On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artif. Intell.* 108(1-2), 69–123 (1999)
22. Rosenberg, I.G.: Minimal clones I: the five types. *Lectures in Universal Algebra (Proc. Conf. Szeged, 1983)*, *Colloq. Math. Soc. J. Bolyai* 43, 405–427 (1986)
23. Schmerl, J.H.: Countable homogeneous partially ordered sets. *Algebra Universalis* 9, 317–321 (1979)

# Affine Systems of Equations and Counting Infinitary Logic\*

Albert Atserias<sup>1,\*\*</sup>, Andrei Bulatov<sup>2</sup>, and Anuj Dawar<sup>3</sup>

<sup>1</sup> Universitat Politècnica de Catalunya, Barcelona, Spain

<sup>2</sup> Simon Fraser University, Burnaby BC, Canada

<sup>3</sup> University of Cambridge Computer Laboratory, Cambridge, UK

**Abstract.** We study the definability of constraint satisfaction problems (CSP) in various fixed-point and infinitary logics. We show that testing the solvability of systems of equations over a finite Abelian group, a tractable CSP that was previously known not to be definable in Datalog, is not definable in an infinitary logic with counting and hence that it is not definable in least fixed point logic or its extension with counting. We relate definability of CSPs to their classification obtained from tame congruence theory of the varieties generated by the algebra of polymorphisms of the template structure. In particular, we show that if this variety admits either the unary or affine type, the corresponding CSP is not definable in the infinitary logic with counting. We also study the complexity of determining whether a CSP omits unary and affine types.

## 1 Introduction

The classification of constraint satisfaction problems (CSP) according to their tractability has been a major research goal since Feder and Vardi first formulated their dichotomy conjecture [1]. The general form of the *constraint satisfaction problem* takes as instance two finite relational structures  $\mathbf{A}$  and  $\mathbf{B}$  and asks if there is a homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ . We think of the elements of  $\mathbf{A}$  as the variables of the problem and the universe of  $\mathbf{B}$  as the domain of values which these variables may take. The individual tuples in the relations of  $\mathbf{A}$  act as constraints on the values that must be matched to the relations holding in  $\mathbf{B}$ . The general form of the problem is NP-complete. In this paper we are mainly concerned with the non-uniform version of the problem which gives rise, for each fixed finite structure  $\mathbf{B}$  to a different decision problem that we denote  $\text{CSP}(\mathbf{B})$ , namely the problem of deciding whether a given  $\mathbf{A}$  maps homomorphically to  $\mathbf{B}$ . For many fixed  $\mathbf{B}$ , this problem is solvable in polynomial time, while for others it remains NP-complete.

In the present paper we are concerned with classifying constraint satisfaction problems according to their definability in a suitable logic. This is an approach that has proved useful in studying the tractability of constraint satisfaction problems [1,2,3]. In particular, it is known that many natural constraint satisfaction

---

\* Research supported by the Isaac Newton Institute, LAA programme.

\*\* Supported in part by CICYT TIN2004-04343.

problems that are tractable are definable (or, to be precise, their complements are definable) in Datalog, the language of function-free Horn clauses. Any class of structures that is definable in Datalog is necessarily decidable in polynomial time, but there are known constraint satisfaction problems that are tractable but are not definable in Datalog. A classical example is the solvability of systems of linear equations over the two-element field [1], which we denote  $\text{CSP}(\mathbb{Z}_2)$ . Bulatov [4] (see also [5]) provides a uniform explanation for the tractability of these by showing that any constraint language that has a Mal'tsev polymorphism is solvable in polynomial time. Furthermore, there are NP-complete constraint satisfaction problems, such as 3-colourability of graphs, which we can show are not Datalog-definable, without requiring the assumption that P is different from NP. Indeed, the class of constraint satisfaction problems whose complements are definable in Datalog appears to be a robust, natural class of problems with many independent and equivalent characterisations [6,7].

A natural question that arises is whether we can offer any explanation based on logical definability for the tractability of problems such as the satisfiability of systems of linear equations over a finite field. Is there a natural logic such that all problems definable in this logic are polynomial-time decidable and that can express  $\text{CSP}(\mathbb{Z}_2)$ ? In particular, is this problem definable in LFP—the extension of first-order logic with least fixed points or  $\text{LFP} + \text{C}$ —the extension of LFP with counting? Both these logics have been extensively studied in the context of descriptive complexity as characterising natural fragments of polynomial time. Interestingly, Blass et al. [8] proved that  $\text{LFP} + \text{C}$  is able to define the class of non-singular square matrices over any fixed finite field, so it would not be very surprising if this logic were able to express  $\text{CSP}(\mathbb{Z}_2)$ . Despite this, it is a consequence of our results that neither of these logics is able to express the solvability of systems of linear equations over any finite field. Indeed, we show that these problems are not definable in  $C_{\infty\omega}^\omega$ , the infinitary logic with bounded number of variables and counting, a logic much more expressive than  $\text{LFP} + \text{C}$ . Combined with the result of Blass, Gurevich and Shelah about non-singular matrices, our result exhibits a fine-grained distinction between the problem of computing the rank of a square matrix and the problem of computing its determinant.

Another important means of classifying constraint satisfaction problems is on the basis of the algebra of the template structure  $\mathbf{B}$ . A polymorphism of a structure is an operation on its universe that preserves all its relations (see Section 2 for precise definitions). It is known that whether or not  $\text{CSP}(\mathbf{B})$  is tractable depends only on the algebra  $\mathcal{B}$  obtained from the universe of  $\mathbf{B}$  endowed with its polymorphisms. Indeed, it depends only on the variety generated by this algebra. This is established in [9] by showing that if the algebra  $\mathcal{B}'$  of structure  $\mathbf{B}'$  is obtained from  $\mathcal{B}$  as a power, subalgebra or homomorphic image, then  $\text{CSP}(\mathbf{B}')$  is polynomial-time reducible to  $\text{CSP}(\mathbf{B})$ . We show in the present paper that this can be improved to Datalog-definable reductions. These are weak reductions that, in particular, preserve definability in LFP and  $C_{\infty\omega}^\omega$ . This allows us to establish that definability of a CSP in these logics is also determined by  $\text{var}(\mathcal{B})$ , the variety generated by the algebra of  $\mathbf{B}$ .



Using the tool of Datalog-reductions, which we expect to be useful for other applications in the area, we relate definability of constraint satisfaction problems in  $C_{\infty\omega}^\omega$  to the classification of varieties of finite algebras from tame congruence theory [10]. It is known [9] that  $\text{CSP}(\mathbf{B})$  is NP-complete if  $\text{var}(\mathbf{B})$  admits the unary type (also known as type 1), and it is conjectured that  $\text{CSP}(\mathbf{B})$  is in P otherwise. Similarly, Larose and Zádori showed [11] that  $\text{CSP}(\mathbf{B})$  is not definable in Datalog if  $\text{var}(\mathbf{B})$  admits the unary or affine types (types 1 and 2), and conjectured the converse. It is a consequence of our results that we can strengthen the assertion by replacing Datalog with  $C_{\infty\omega}^\omega$ . This implies that, if the Larose-Zádori conjecture is true, we obtain a dichotomy of definability whereby, for every  $\mathbf{B}$ , either  $\text{CSP}(\mathbf{B})$  is definable in Datalog or it is not definable in  $C_{\infty\omega}^\omega$ .

Finally, we consider the meta-problems of deciding, given a structure  $\mathbf{B}$  or an algebra  $\mathcal{B}$  whether or not  $\text{var}(\mathcal{B})$  omits the unary and affine types. For algebras, the problem was shown decidable in polynomial time in [12], while for structures we show it is NP-complete.

## 2 Preliminaries

*Structures and graphs* A vocabulary  $\sigma$  is a finite collection of relation symbols, each with an associated arity. A  $\sigma$ -structure  $\mathbf{A}$  consists of a finite set  $A$  with a relation  $R^{\mathbf{A}} \subseteq A^r$  for each  $r$ -ary relation symbol  $R$  in  $\sigma$ . A graph is a structure with a binary relation that is symmetric and irreflexive. A *homomorphism* from a  $\sigma$ -structure  $\mathbf{A}$  to a  $\sigma$ -structure  $\mathbf{B}$  is a map  $h : A \rightarrow B$  such that for each  $R$  in  $\sigma$  and each  $\mathbf{a} \in A^r$ , if  $\mathbf{a} \in R^{\mathbf{A}}$  then  $h(\mathbf{a}) \in R^{\mathbf{B}}$ . We write  $\mathbf{A} \rightarrow \mathbf{B}$  to denote that there exists a homomorphism from  $\mathbf{A}$  to  $\mathbf{B}$ . We write  $\text{CSP}(\mathbf{B})$  for the class of finite structures  $\mathbf{A}$  such that  $\mathbf{A} \rightarrow \mathbf{B}$  and also for the decision problem of determining membership in this class.

For the standard definition of the treewidth of a graph, we refer the reader to [13]. In our proofs we will use the following alternative characterization in terms of the *cops and robber game* [14]. The game is played by two players, one of whom controls the set of  $k$  cops attempting to catch a robber controlled by the other player. The cop player can move any set of cops to any vertices of the graph, while the robber can *simultaneously* move along any path in the graph as long as there is no cop currently on the path. It is known [14] that the cop player has a winning strategy on a graph using  $k + 1$  cops if and only if the graph has treewidth at most  $k$ . The treewidth of a graph  $G$  is denoted  $tw(G)$ .

*Logic.* A formula is *positive quantifier-free* if it is formed from the atomic formulas using conjunctions and disjunctions. A formula is *existential positive* if it is formed from the atomic formulas using conjunctions, disjunctions and existential quantification. Datalog is the extension of *existential positive formulas* with a recursion mechanism. Similarly, LFP is the extension of *full first-order logic* with an operator for forming the least fixed points of positive formulas. Finally, LFP + C is the extension of LFP with a counting mechanism. For formal definitions, which we will not need in this paper, we refer the reader to [15].

It is known that every class of structures definable in  $LFP + C$  is decidable in polynomial time.

The formulas of the logic  $C_{\infty\omega}$  are obtained from the atomic formulas using negation, infinitary conjunction and disjunction, and counting quantifiers ( $\exists^i x\phi$  for any integer  $i \geq 0$ ). The fragment  $C_{\infty\omega}^k$  consists of those formulas of  $C_{\infty\omega}$  in which only  $k$  distinct variables appear and  $C_{\infty\omega}^\omega = \bigcup_{k \in \omega} C_{\infty\omega}^k$ . The significance of  $C_{\infty\omega}^\omega$  is that fixed-point logics can be translated into it. That is, any formula of Datalog or LFP, and indeed of  $LFP + C$  is equivalent to one of  $C_{\infty\omega}^\omega$ . Moreover, these translations into infinitary logics have provided some of the most effective tools for proving inexpressibility results for the fixed-point logics. See [16,17] for a discussion of this and the role of these logics in descriptive complexity.

The expressive power of  $C_{\infty\omega}^\omega$  is characterised by a game known as the *bijective game* [18]. This is played by two players, Spoiler and Duplicator, on a pair of structures  $\mathbf{A}$  and  $\mathbf{B}$ , with  $k$  pairs of pebbles  $(x_i, y_i)$  for  $1 \leq i \leq k$ . At each stage of the game, some of the pebbles may be on elements of the structures with  $x_i$  on an element of  $A$  and  $y_i$  on an element of  $B$ . We write  $a_i$  for the element currently pebbled by  $x_i$ , and  $b_i$  for the element pebbled by  $y_i$ . For each move, Spoiler chooses a pair of pebbles  $(x_i, y_i)$ , Duplicator chooses a bijection  $f : A \rightarrow B$  such that  $f(a_j) = b_j$  for  $i \neq j$ , and Spoiler chooses  $a \in A$  and places  $x_i$  on  $a$  and  $y_i$  on  $f(a)$ . If, after some move, the map  $a_i \mapsto b_i (1 \leq i \leq k)$  is not a partial isomorphism, Spoiler wins; Duplicator wins infinite plays. By a result of Hella [18], Duplicator has a winning strategy if, and only if,  $\mathbf{A}$  and  $\mathbf{B}$  cannot be distinguished by any formula of  $C_{\infty\omega}^k$ , a fact denoted by  $\mathbf{A} \equiv^{C^k} \mathbf{B}$ .

*Universal algebra.* An  $n$ -ary operation  $f$  on a set  $A$  is a *polymorphism* of a relation  $R \subseteq A^n$  if, for any tuples  $\mathbf{a}_1, \dots, \mathbf{a}_n \in R$ , the  $r$ -tuple obtained by applying  $f$  component-wise also belongs to  $R$ . We say that  $R$  is invariant under  $f$ .

A set with a collection of operations on it is called an *algebra*. Every structure  $\mathbf{A}$  can be naturally associated with an algebra  $Al(\mathbf{A})$ , called the *algebra* of  $\mathbf{A}$ , whose base set is the universe of  $\mathbf{A}$ , and whose operations are the polymorphisms of  $\mathbf{A}$ . A *variety* is a class of algebras which, if it contains  $\mathcal{A}$  also contains every subalgebra of  $\mathcal{A}$ , every homomorphic image of  $\mathcal{A}$ , and every direct power of  $\mathcal{A}$ . The smallest variety containing  $\mathcal{A}$  is called the *variety generated by  $\mathcal{A}$*  and denoted by  $var(\mathcal{A})$ . For further background on universal algebra, see [19].

### 3 Definability of Equations

In this section we show that the problem of determining the solvability of linear equations over the two-element field, which we mentioned above as a canonical example of a tractable CSP whose complement is not definable in Datalog, is also not definable in  $C_{\infty\omega}^\omega$ . Indeed, we prove a more general result by showing that the solvability of equations over a finite Abelian group  $\mathcal{G}$  with at least two elements is not definable in  $C_{\infty\omega}^\omega$ . In the following we will write  $+$  for the group operation in  $\mathcal{G}$  and  $0$  for the identity.

Consider the following formulation of the problem.

**Definition 1.** Let  $\mathcal{G}$  be a finite Abelian group over a set  $G$  and  $r$  be a positive integer. We define the structure  $\mathbf{E}_{\mathcal{G},r}$  to have universe  $G$  and, for each  $a \in G$  and  $1 \leq j \leq r$ , it has a relation  $R_a^j$  of arity  $j$  that consists of the set of tuples  $(x_1, \dots, x_j) \in G^j$  that satisfy the equation  $x_1 + \dots + x_j = a$ .

Thus, any structure  $\mathbf{A}$  in the signature of  $\mathbf{E}_{\mathcal{G},r}$  can be seen as a set of equations in which at most  $r$  variables occur in each equation. The universe of  $\mathbf{A}$  is the set of variables and the occurrence of a tuple  $(x_1, \dots, x_j)$  in a relation  $R_a^j$  signifies the equation  $x_1 + \dots + x_j = a$ . This set of equations is solvable if, and only if,  $\mathbf{A} \rightarrow \mathbf{E}_{\mathcal{G},r}$ . In the sequel we will say “the equation  $x_1 + \dots + x_j = a$  occurs in  $\mathbf{A}$ ” to mean that the tuple  $(x_1, \dots, x_j)$  is in  $R_a^j$ .

Our aim now is to exhibit, for each non-trivial finite Abelian group  $\mathcal{G}$  and each positive integer  $k$ , a pair of structures  $\mathbf{A}$  and  $\mathbf{B}$  such that  $\mathbf{A} \equiv^{C^k} \mathbf{B}$  and such that  $\mathbf{A} \in \text{CSP}(\mathbf{E}_{\mathcal{G},3})$  and  $\mathbf{B} \notin \text{CSP}(\mathbf{E}_{\mathcal{G},3})$ . This will show that  $\text{CSP}(\mathbf{E}_{\mathcal{G},3})$  is not definable in  $C_{\infty\omega}^\omega$ . This, of course, implies the result for all  $\text{CSP}(\mathbf{E}_{\mathcal{G},r})$  with  $r \geq 3$ . The structures we construct are sets of equations derived from 3-regular graphs of large treewidth. From now on, fix a non-trivial finite Abelian group  $\mathcal{G}$  over a set  $G$ , a 3-regular graph  $H$ , and a distinguished vertex  $u$  of  $H$ . We define, for each  $a \in G$ , a set of equations  $\mathbf{E}_a H^u$  as follows (note that  $\mathbf{E}_a H^u$  is a structure over the vocabulary of  $\mathbf{E}_{\mathcal{G},3}$ ):

For each vertex  $v \in V^H$  and each edge  $e \in E^H$  that is incident on  $v$ , we have  $m$  distinct variables  $x_i^{v,e}$  where  $i$  ranges over  $G$ . Since each vertex has three edges incident on it, there are  $3m$  variables associated to each vertex. For every vertex  $v$  other than  $u$ , let  $e_1, e_2, e_3$  be the three edges incident on  $v$ . We then include the following equation in  $\mathbf{E}_a H^u$  for all  $i, j, k \in G$ :

$$x_i^{v,e_1} + x_j^{v,e_2} + x_k^{v,e_3} = i + j + k. \tag{1}$$

For the distinguished vertex  $u$ , instead of the above, we include the following equation, again for all  $i, j, k \in G$ :

$$x_i^{u,e_1} + x_j^{u,e_2} + x_k^{u,e_3} = i + j + k + a. \tag{2}$$

In addition, for each edge  $e \in E^H$  let  $v_1, v_2$  be its endpoints. We include the following equations in  $\mathbf{E}_a H^u$  for all  $i, j \in G$ :

$$x_i^{v_1,e} + x_j^{v_2,e} = i + j. \tag{3}$$

We refer to equations of the form (1) and (2) as *vertex equations* and equations of the form (3) as *edge equations*.

**Lemma 2.**  $\mathbf{E}_a H^u$  is satisfiable if, and only if,  $a = 0$

*Proof.* To see that  $\mathbf{E}_0 H^u$  is satisfiable, just take the assignment that gives the variable  $x_i^{v,e}$  the value  $i$ . To see that  $\mathbf{E}_a H^u$  is unsatisfiable when  $a \neq 0$ , consider the subsystem  $S_0$  of equations involving only the variables  $x_0^{v,e}$  with subscript 0. Note that each such variable occurs exactly twice in  $S_0$ , once in a vertex equation and once in an edge equation. Thus, if we add up the left hand sides

of all equations in  $S_0$ , we get  $2 \sum x_0^{v,e}$ . Note also that each variable  $x_0^{v,e}$  has a companion variable  $x_0^{v',e}$  where  $v'$  is the other endpoint of the edge  $e$  and we have the equation  $x_0^{v,e} + x_0^{v',e} = 0$ . Thus  $2 \sum_{v,e} x_0^{v,e} = 2 \sum_e (x_0^{v,e} + x_0^{v',e}) = 0$ . On the other hand, the right-hand side of all equations is 0 except for the one vertex equation for  $u$ , which has right-hand side  $a$ . Thus summing the right-hand sides of all equations gives the sum  $a$ . Since  $a \neq 0$ , this shows that the subsystem  $S_0$  and hence the system of equations  $\mathbf{E}_a H^u$  is unsatisfiable.

**Lemma 3.** *If  $\text{tw}(H) > k$  and  $H$  is connected, then  $\mathbf{E}_0 H^u \equiv^{C^k} \mathbf{E}_a H^u$  for any  $a \in \mathcal{G}$ .*

*Proof.* Our aim is to exhibit a winning strategy for Duplicator in the  $k$ -pebble bijective game played on the two structures  $\mathbf{A} = \mathbf{E}_0 H^u$  and  $\mathbf{B} = \mathbf{E}_a H^u$ . Since  $\text{tw}(H) > k$ , we know that in the  $k$  cops and robber game played on  $H$ , robber has a winning strategy and Duplicator will make use of this strategy.

For each vertex  $v \in V^H$  let  $X^v$  denote the set of variables  $x_i^{v,e}$  for edges  $e$  incident on  $v$ . Similarly, for each  $e \in E^H$ , let  $X^e$  denote the set of variables involving  $e$ .

We say that a bijection  $f : \mathbf{A} \rightarrow \mathbf{B}$  is *good* for a vertex  $v \in V^H$  if the following conditions hold:

1. for all  $w \in V^H$ ,  $fX^w = X^w$ ;
2. for all  $e \in E^H$ ,  $fX^e = X^e$ ;
3. for all  $x, y$ , if  $x + y = i$  is an equation in  $\mathbf{A}$  then  $f(x) + f(y) = i$  is an equation in  $\mathbf{B}$ ; and
4. for all  $x, y, z$ , if  $x + y + z = i$  is an equation in  $\mathbf{A}$ , then
  - $f(x) + f(y) + f(z) = i$  is an equation in  $\mathbf{B}$  if  $x, y, z \notin X^v$ ; and
  - $f(x) + f(y) + f(z) = i + a$  is an equation in  $\mathbf{B}$  if  $x, y, z \in X^v$ .

Note that the identity is a bijection that is good for  $u$ . Also, note that a bijection that is good for  $v$  preserves all equations except the vertex equations for  $v$ .

*Claim.* Given a bijection  $f : \mathbf{A} \rightarrow \mathbf{B}$  that is good for  $v$ , if there is a path in  $H$  from  $v$  to  $w$  avoiding  $u_1, \dots, u_k$  then there is a bijection  $f' : \mathbf{A} \rightarrow \mathbf{B}$  that is good for  $w$  such that  $f|_{(X^{u_1} \cup \dots \cup X^{u_k})} = f'|_{(X^{u_1} \cup \dots \cup X^{u_k})}$ .

*Proof.* Let the path from  $v$  to  $w$  avoiding  $u_1, \dots, u_k$  be  $v = v_1, \dots, v_n = w$ . For each edge  $e = \{v_i, v_{i+1}\}$  along this path, write  $x_j^{e-}$  for the variable  $x_j^{v_i,e}$  and  $x_j^{e+}$  for the variable  $x_j^{v_{i+1},e}$ . We then define  $f'$  by  $f'(x_j^{e-}) = f(x_{j-a}^{e-})$  and  $f'(x_j^{e+}) = f(x_{j+a}^{e+})$ ; and  $f'$  agrees with  $f$  everywhere else. In particular, since the path from  $v$  to  $w$  avoids  $u_1, \dots, u_k$ ,  $f'$  agrees with  $f$  on  $X^{u_1} \cup \dots \cup X^{u_k}$ .

We now describe Duplicator’s winning strategy in the bijective  $k$ -pebble game. Duplicator responds to Spoiler’s first move with the identity bijection. She maintains a board on the side which describes a position in the  $k$  cops and robber game played on the graph  $H$ . At any point, if Spoiler’s pebbles are on the position  $x_1, \dots, x_k$  in  $\mathbf{A}$  and  $v_1, \dots, v_k$  are the vertices of  $H$  to which these variables

correspond, then the current position of the cops and robber game has  $k$  cops on the vertices  $v_1, \dots, v_k$ . If the robber’s position according to its winning strategy is  $v$ , then Duplicator will play a bijection that is good for  $v$ .

To see that Duplicator can do this forever, suppose Spoiler lifts a pebble from  $x_i$ . Duplicator responds with a current bijection  $f$  that is good for  $v$ . Since the only equations not preserved by  $f$  are those associated with the vertex  $v$ , Spoiler must place at least three pebbles on variables associated with  $v$  to win the game. However, Duplicator responds to Spoiler placing the pebble on a new position  $x'_i$  by updating the position of the cops and robber game. Suppose robber’s winning strategy dictates that the robber move from  $v$  to  $w$ . Since robber’s move must be along a path avoiding the current cop positions, by Claim 3, Duplicator can update the bijection  $f$  to an  $f'$  that is good for  $w$  and agrees with  $f$  on all currently pebbled positions. It is now clear that Duplicator can play forever.

**Theorem 4.** *Let  $\mathcal{G}$  be a non-trivial finite Abelian group. Then  $\text{CSP}(\mathbf{E}_{\mathcal{G},3})$  is not definable in  $C_{\infty\omega}^\omega$*

*Proof.* Suppose, to the contrary, that there is a  $k$  such that  $\text{CSP}(\mathbf{E}_{\mathcal{G},3})$  is definable in  $C_{\infty\omega}^k$ . Let  $H$  be any connected, 3-regular graph with  $\text{tw}(H) > k$  and  $u$  any vertex of  $H$ . For instance,  $H$  could be a sufficiently large brick graph. Let  $a$  be any element of  $\mathcal{G}$  distinct from 0. Then, by Lemma 2,  $\mathbf{E}_0 H^u \in \text{CSP}(\mathbf{E}_{\mathcal{G},3})$  and  $\mathbf{E}_a H^u \notin \text{CSP}(\mathbf{E}_{\mathcal{G},3})$ . But, by Lemma 3,  $\mathbf{E}_0 H^u \equiv^{C^k} \mathbf{E}_a H^u$ , a contradiction.

## 4 Logical Reductions

### 4.1 Definition

Let  $\sigma$  and  $\tau = (R_1, \dots, R_s)$  be two relational vocabularies. A  $k$ -ary interpretation with  $p$  parameters of  $\tau$  in  $\sigma$  is an  $(s + 1)$ -tuple  $\mathbf{I} = (\varphi_U, \varphi_1, \dots, \varphi_s)$  of formulas over the vocabulary  $\tau$ , where  $\varphi_U = \varphi_U(\mathbf{x}, \mathbf{y})$  has  $k + p$  free variables  $\mathbf{x} = (x^1, \dots, x^k)$  and  $\mathbf{y} = (y_1, \dots, y_p)$ , and  $\varphi_i = \varphi_i(\mathbf{x}_1, \dots, \mathbf{x}_r, \mathbf{y})$  has  $kr$  free variables where  $r$  is the arity of  $R_i$  and each  $\mathbf{x}_j = (x_j^1, \dots, x_j^k)$  and  $\mathbf{y} = (y_1, \dots, y_p)$ .

Let  $\mathbf{A}$  be a  $\sigma$ -structure. A tuple  $\mathbf{c} = (a_1, \dots, a_p)$  of pairwise different points of  $\mathbf{A}$  is called proper. The interpretation of  $\mathbf{A}$  through  $\mathbf{I}$  with parameters  $\mathbf{c}$ , denoted by  $\mathbf{I}(\mathbf{A}, \mathbf{c})$ , is the  $\tau$ -structure whose universe is  $\{\mathbf{a} \in A^k : \mathbf{A} \models \varphi_U(\mathbf{a}, \mathbf{c})\}$ , and whose interpretation for  $R_i$  is the set of tuples  $(\mathbf{a}_1, \dots, \mathbf{a}_r) \in (A^k)^r$  such that  $\mathbf{A} \models \varphi_U(\mathbf{a}_1, \mathbf{c}) \wedge \dots \wedge \varphi_U(\mathbf{a}_r, \mathbf{c}) \wedge \varphi_i(\mathbf{a}_1, \dots, \mathbf{a}_r, \mathbf{c})$ . If each formula in  $\mathbf{I}$  belongs to a class of formulas  $\Theta$ , we say that  $\mathbf{I}$  is a  $\Theta$ -interpretation.

Now we are ready to define the notion of logical reduction:

**Definition 5.** *Let  $\mathcal{C}$  be a class of  $\sigma$ -structures,  $\mathcal{D}$  a class of  $\tau$ -structures closed under isomorphisms, and  $\Theta$  be a class of formulas. A  $\Theta$ -interpretation with  $p$  parameters  $\mathbf{I}$  of  $\tau$  in  $\sigma$  is a  $\Theta$ -reduction from  $\mathcal{C}$  to  $\mathcal{D}$  if, for every  $\sigma$ -structure  $\mathbf{A}$  with at least  $p$  elements,  $\mathbf{A} \in \mathcal{C}$  if, and only if,  $\mathbf{I}(\mathbf{A}, \mathbf{c}) \in \mathcal{D}$  for some proper  $\mathbf{c}$ .*

If such a reduction exists, we say that  $\mathcal{C}$  reduces to  $\mathcal{D}$  under  $\Theta$ -reductions, and write  $\mathcal{C} \leq_\Theta \mathcal{D}$ . We use the collections of positive quantifier-free formulas, existential positive formulas, and datalog formulas (i.e. datalog programs) and write

$\leq_{\text{pqf}}$ ,  $\leq_{\text{ep}}$  and  $\leq_{\text{datalog}}$ , respectively. These are reductions of increasing power, and definability in  $C_{\infty\omega}^\omega$  is preserved downwards by all three.

### 4.2 Expansions by Reduced Invariant Relations

Let  $A$  be a set and let  $R \subseteq A^s$  be a relation on  $A$ . We define an equivalence relation  $\theta(R)$  on  $\{1, \dots, s\}$  by setting  $(i, j) \in \theta(R)$  if, and only if,  $a_i = a_j$  for every  $(a_1, \dots, a_s) \in R$ . We say that  $R$  is *reduced* if  $\theta(R)$  is the trivial equivalence relation (i.e. equality). Note that the equality relation on  $A$  is not reduced. The proof of the following lemma appears in the full version of this paper.

**Lemma 6.** *Let  $\mathbf{B}$  be a finite structure, and  $\mathbf{D}$  be an expansion of  $\mathbf{B}$  by a reduced relation invariant under all polymorphisms of  $\mathbf{B}$ . Then,  $\text{CSP}(\mathbf{D}) \leq_{\text{pqf}} \text{CSP}(\mathbf{B})$ .*

Next we show that reduced relations are general enough. First a piece of notation: Let  $\mathbf{a} = (a_1, \dots, a_m)$  be a sequence and let  $I = (i_1, \dots, i_r)$  be a sequence of indices, where  $1 \leq i_j \leq m$  for every  $j \in \{1, \dots, r\}$ . We write  $\mathbf{a}_I$  for the sequence  $(a_{i_1}, \dots, a_{i_r})$ . Now let  $R$  be a relation of arity  $s$  and  $I$  a sequence of indices from  $\{1, \dots, s\}$ . Then  $\text{pr}_I R$  denotes the relation  $\{\mathbf{a}_I : \mathbf{a} \in R\}$ .

Let  $R$  be a relation of arity  $s$  and  $\theta(R)$ , the equivalence relation on  $\{1, \dots, s\}$  as defined in the previous section. Let  $I$  be a set of representatives of the equivalence-classes of  $\theta(R)$ , ordered in an arbitrary way, and define  $\text{red}(R) = \text{pr}_I R$ . Note that  $\text{red}(R)$  does not depend on the choice of  $I$ . Besides, for every  $i \notin I$  there exists some  $j \in I$  such that  $a_i = a_j$  for every tuple  $(a_1, \dots, a_s) \in R$ . We call  $\text{red}(R)$  the reduced version of  $R$ . A reduced structure is a structure all whose relations are reduced. To every structure  $\mathbf{B}$  we can associate a reduced structure, called the *reduced version of  $\mathbf{B}$* , whose universe is the universe of  $\mathbf{B}$  itself and whose relations are the reduced versions of the relations of  $\mathbf{B}$ . Note that the vocabularies of a structure and its reduced version may be different. Note that the polymorphisms of  $\mathbf{B}$  and its reduced version are the same.

**Lemma 7.** *Let  $\mathbf{B}$  a finite structure and let  $\mathbf{D}$  be the reduced version of  $\mathbf{B}$ . Then  $\text{CSP}(\mathbf{B}) \leq_{\text{datalog}} \text{CSP}(\mathbf{D})$  and  $\text{CSP}(\mathbf{D}) \leq_{\text{pqf}} \text{CSP}(\mathbf{B})$ .*

*Proof:* For space constraints, we only sketch the reduction  $\text{CSP}(\mathbf{B}) \leq_{\text{datalog}} \text{CSP}(\mathbf{D})$ . Let  $\mathbf{A}$  be an instance of  $\text{CSP}(\mathbf{B})$ . We define an instance  $\mathbf{C}$  of  $\text{CSP}(\mathbf{D})$ . The universe of  $\mathbf{C}$  is  $A$  itself. For the relations, the basic idea is to project every relation  $R^{\mathbf{A}}$  to the coordinates of a set of representatives  $I$  of the  $\theta$ -classes, where  $\theta = \theta(R)$ . However, before we do that, we need to *close* each  $R^{\mathbf{A}}$  under all equalities implied by the equivalences  $(i, j) \in \theta$ . We do that using Datalog-definable intermediate relations.

Let  $E$  be the binary relation on  $A$  defined by the following Datalog program:

$$\begin{aligned} E(x_i, x_j) &: - R(x_1, \dots, x_s) \\ E(x, y) &: - E(y, x) \\ E(x, z) &: - E(x, y) \wedge E(y, z), \end{aligned}$$

where the first rule is introduced for every symbol  $R$  in  $\sigma$  and every  $(i, j) \in \theta(R)$ . It is obvious that  $E$  is an equivalence relation on  $A$ ; reflexivity follows from the fact that  $(i, j) \in \theta(R)$  in the first rule, symmetry is enforced by the second rule, and transitivity is enforced by the third. Next, for every  $r$ -ary symbol  $R$  in  $\sigma$ , let  $R'$  be the relation defined by  $R'(\mathbf{x}_I) :- R(y_1, \dots, y_s) \wedge E(x_1, y_1) \wedge \dots \wedge E(x_s, y_s)$ , where  $I$  is a set of representatives of the  $\theta(R)$ -classes ordered in an arbitrary way. This defines  $\mathbf{C}$ , and we defined it by a Datalog program interpreted on  $\mathbf{A}$ . It remains to argue that this datalog-interpretation is indeed a reduction. The argument may be found in the full version of this paper.  $\square$

### 4.3 Powering, Subalgebras, and Homomorphic Images

In this subsection we show how the basic algebraic constructions of powering, subalgebra and homomorphic images can be handled by Datalog-reductions. In the following, fix a finite structure  $\mathbf{B}$  and its corresponding algebra  $\mathcal{B}$ .

Suppose  $\mathcal{B}'$  is an algebra that has a homomorphic image  $\mathcal{A} = h(\mathcal{B}')$  that is a reduct of  $\mathcal{B}$ . Note that  $A = B = h(B')$ , i.e. the universes of  $\mathcal{A}$  and  $\mathcal{B}$  are the same and are the image of the universe of  $\mathcal{B}'$  under  $h$ . We define a new structure  $\mathbf{B}' = \text{pre}(\mathbf{B}, h)$ , the *preimage* of  $\mathbf{B}$ , whose universe is  $B'$  and whose relations are the preimages  $h^{-1}(R^{\mathbf{B}})$  of the relations  $R^{\mathbf{B}}$  of  $\mathbf{B}$ .

**Lemma 8.** *Let the algebras  $\mathcal{B}$  and  $\mathcal{B}'$ , and the structures  $\mathbf{B}$  and  $\mathbf{B}' = \text{pre}(\mathbf{B}, h)$  be as above. Then  $\text{CSP}(\mathbf{B}) \leq_{\text{pqf}} \text{CSP}(\mathbf{B}')$  and  $\mathcal{B}'$  is a reduct of  $\text{Al}(\mathbf{B}')$ .*

Suppose  $\mathcal{B}'$  is an algebra that has a subalgebra  $\mathcal{A} \subseteq \mathcal{B}'$  that is a reduct of  $\mathcal{B}$ . Note that  $A = B \subseteq B'$ , i.e. the universes of  $\mathcal{A}$  and  $\mathcal{B}$  are the same and are a subset of the universe of  $\mathcal{B}'$ . We define a new structure  $\mathbf{B}' = \text{ext}(\mathbf{B}, B')$ , the *extension* of  $\mathbf{B}$ , with universe  $B'$  and the same relations as  $\mathbf{B}$ .

**Lemma 9.** *Let the algebras  $\mathcal{B}$  and  $\mathcal{B}'$ , and the structures  $\mathbf{B}$  and  $\mathbf{B}' = \text{ext}(\mathbf{B}, B')$  be as above. Then  $\text{CSP}(\mathbf{B}) \leq_{\text{pqf}} \text{CSP}(\mathbf{B}')$  and  $\mathcal{B}'$  is a reduct of  $\text{Al}(\mathbf{B}')$ .*

Let  $R$  be an  $r$ -ary relation on the set  $A^n$ . Then the *flattening* of  $R$ , denoted  $\text{fla}(R, n)$ , is the  $rn$ -ary relation on  $A$  that contains all tuples  $(x_1, \dots, x_{rn})$  such that  $((x_1, \dots, x_n), \dots, (x_{(r-1)n+1}, \dots, x_{rn})) \in R$ . Suppose  $\mathcal{B}'$  is an algebra that has a direct power  $\mathcal{A} = \mathcal{B}'^n$  that is a reduct of  $\mathcal{B}$ . Note that  $A = B = B'^n$ , i.e. the universes of  $\mathcal{A}$  and  $\mathcal{B}$  are the same and are the  $n$ -th power of the universe of  $\mathcal{B}'$ . We define a new structure  $\mathbf{B}' = \text{fla}(\mathbf{B}, n)$ , the *flattening* of  $\mathbf{B}$ , whose universe is  $B$  and whose relations are the flattenings of the relations of  $\mathbf{B}$ .

**Lemma 10.** *Let the algebras  $\mathcal{B}$  and  $\mathcal{B}'$ , and the structures  $\mathbf{B}$  and  $\mathbf{B}' = \text{fla}(\mathbf{B}, n)$  be as above. Then  $\text{CSP}(\mathbf{B}) \leq_{\text{pqf}} \text{CSP}(\mathbf{B}')$  and  $\mathcal{B}'$  is a reduct of  $\text{Al}(\mathbf{B}')$ .*

These lemmas allow us to derive the following consequence we require.

**Theorem 11.** *Let  $\mathbf{B}$  and  $\mathbf{B}'$  be finite structures and let  $\mathcal{B}$  and  $\mathcal{B}'$  be their respective algebras. If  $\text{var}(\mathcal{B}')$  contains a reduct of  $\mathcal{B}$ , then  $\text{CSP}(\mathbf{B}) \leq_{\text{datalog}} \text{CSP}(\mathbf{B}')$ .*

Proof: Suppose that some algebra  $\mathcal{A}$  of  $\text{var}(\mathcal{B}')$  is a reduct of  $\mathcal{B}$ . By the HSP-theorem [19, Theorem 9.5]  $\mathcal{A}$  is a homomorphic image of a subalgebra of a direct power of  $\mathcal{B}'$ . Let  $\mathcal{B}_p$ ,  $\mathcal{B}_s$ , and  $\mathcal{B}_h$  be the direct power, its subalgebra, and the homomorphic image, respectively. We have  $\mathcal{A} = \mathcal{B}_h$ . Let  $n$  be such that  $\mathcal{B}_p = \mathcal{B}'^n$ , and let  $h$  be a homomorphism from  $\mathcal{B}_s$  to  $\mathcal{B}_h$ .

We use three intermediate structures  $\mathbf{B}_s = \text{pre}(\mathbf{B}, h)$ ,  $\mathbf{B}_p = \text{ext}(\mathbf{B}_s, B_p)$ , and  $\mathbf{B}_f = \text{fla}(\mathbf{B}_p, n)$  that, by the definition, have the universes of the algebras  $\mathcal{B}_s$ ,  $\mathcal{B}_p$ , and  $\mathcal{B}'$  respectively. By Lemma 8,  $\text{CSP}(\mathbf{B}) \leq_{\text{pqf}} \text{CSP}(\mathbf{B}_s)$  and  $\mathcal{B}_s$  is a reduct of  $\text{Al}(\mathbf{B}_s)$ . By Lemma 9,  $\text{CSP}(\mathbf{B}_s) \leq_{\text{pqf}} \text{CSP}(\mathbf{B}_p)$  and  $\mathcal{B}_p$  is a reduct of  $\text{Al}(\mathbf{B}_p)$ . By Lemma 10,  $\text{CSP}(\mathbf{B}_p) \leq_{\text{pqf}} \text{CSP}(\mathbf{B}_f)$ , and  $\mathcal{B}'$  is a reduct of  $\text{Al}(\mathbf{B}_f)$ . Now, let  $\mathbf{D}$  be the reduced version of  $\mathbf{B}_f$ . Then  $\text{CSP}(\mathbf{B}_f) \leq_{\text{datalog}} \text{CSP}(\mathbf{D})$  by Lemma 7. We prove that  $\text{CSP}(\mathbf{D}) \leq_{\text{pqf}} \text{CSP}(\mathcal{B}')$  and the result will follow by composing.

Let  $\mathbf{D}'$  be the expansion of  $\mathbf{D}$  obtained by adding all the relations of  $\mathcal{B}'$ . Since  $\mathcal{B}'$  is a reduct of the algebra of  $\mathbf{B}_f$  and  $\mathbf{D}'$  is the flattening of  $\mathbf{B}_f$ , it is straightforward that every relation of  $\mathbf{D}'$  is invariant under all polymorphisms of  $\mathcal{B}'$ . Moreover, the relations in  $\mathbf{D}'$  that are not in  $\mathcal{B}'$  are reduced, so  $\text{CSP}(\mathbf{D}') \leq_{\text{pqf}} \text{CSP}(\mathcal{B}')$  by Lemma 6. It is also obvious that  $\text{CSP}(\mathbf{D}) \leq_{\text{pqf}} \text{CSP}(\mathbf{D}')$  through the mapping that sends a structure to its expansion with empty relations. Composing we get  $\text{CSP}(\mathbf{D}) \leq_{\text{pqf}} \text{CSP}(\mathcal{B}')$ .  $\square$

#### 4.4 Reduction from the Idempotent Case

To every finite structure  $\mathbf{B}$  we associate a new structure, the *singleton-expansion* of  $\mathbf{B}$ , by adding one unary relation  $\{b\}$  for every  $b \in B$ . In other words, if  $B = \{b_1, \dots, b_n\}$ , then the structure  $(\mathbf{B}, \{b_1\}, \dots, \{b_n\})$  is the singleton-expansion of  $\mathbf{B}$ . Note that the polymorphisms of the singleton-expansion of  $\mathbf{B}$  are exactly the *idempotent* polymorphisms of  $\mathbf{B}$ , that is polymorphisms  $f$  satisfying the identity  $f(x, \dots, x) = x$ . Indeed, every singleton set  $\{b\}$  is preserved by any idempotent polymorphism of  $\mathbf{B}$ , and any polymorphism of  $\mathbf{B}$  that preserves every singleton set  $\{b\}$  must be idempotent.

**Lemma 12.** *Let  $\mathbf{B}$  be a finite structure, and let  $\mathbf{D}$  be the singleton-expansion of  $\mathbf{B}$ . Then  $\text{CSP}(\mathbf{B}) \leq_{\text{pqf}} \text{CSP}(\mathbf{D})$  and if  $\mathbf{B}$  is a core with at least two points, then  $\text{CSP}(\mathbf{D}) \leq_{\text{ep}} \text{CSP}(\mathbf{B})$ .*

### 5 Omitting Types and Results

Let  $\mathcal{A}$  be an algebra. A *congruence* of  $\mathcal{A}$  is an equivalence relation  $\alpha$  that is invariant with respect to all operations of  $\mathcal{A}$ . In other words, for any ( $n$ -ary) operation  $f$  of  $\mathcal{A}$  and any  $a_1, \dots, a_n, b_1, \dots, b_n \in \mathcal{A}$  such that  $(a_i, b_i) \in \alpha$  we have  $(f(a_1, \dots, a_n), f(b_1, \dots, b_n)) \in \alpha$ . The congruences of  $\mathcal{A}$  form its *congruence lattice*  $\text{con}(\mathcal{A})$ . A *prime quotient* in this lattice is a pair of congruences  $\alpha, \beta$  such that  $\alpha \leq \beta$ ,  $\alpha \neq \beta$ , and for any  $\gamma$  with  $\alpha \leq \gamma \leq \beta$  we have either  $\alpha = \gamma$ , or  $\beta = \gamma$ . The fact that  $\alpha, \beta$  is a prime quotient will be denoted by  $\alpha \prec \beta$ .

Tame congruence theory [10] allows one to assign to each prime quotient of the congruence lattice  $\text{con}(\mathcal{A})$  of a finite algebra  $\mathcal{A}$  one of five types. The type reflects



the local structure of the algebra, which can be one of the following: **1** a finite set with a group action on it (unary type), **2** a finite vector space over a finite field (affine type), **3** a two-element Boolean algebra, **4** a two-element lattice, **5** a two-element semilattice. We use tame congruence as a black box extracting properties we need from existing results, and we do not therefore need a precise definition of the types.

The type of a prime quotient  $\alpha \prec \beta$  is denoted by  $\text{typ}(\alpha, \beta)$ , while  $\text{typ}(\mathcal{A})$  denotes the set of types appearing as types of some prime quotient of  $\mathcal{A}$ . If  $\mathfrak{A}$  is a class of algebras,  $\text{typ}(\mathfrak{A})$  denotes the set  $\bigcup_{\mathcal{A} \in \mathfrak{A}} \text{typ}(\mathcal{A})$ . If  $i \notin \text{typ}(\mathfrak{A})$ , we say that  $\mathfrak{A}$  *omits* type  $i$ . Otherwise, we say  $\mathfrak{A}$  *admits* type  $i$ . We need the following:

**Lemma 13.** *Let  $\mathcal{A}$  be a finite idempotent algebra. If  $\text{var}(\mathcal{A})$  admits types **1** or **2** then it contains a finite idempotent reduct of a module.*

Recall from Section 3 the definition of the structure  $\mathbf{E}_{\mathcal{G},r}$  for every finite Abelian group  $\mathcal{G}$  and every integer  $r \geq 1$ .

**Lemma 14.** *Let  $\mathcal{M}$  be a finite module, let  $\mathcal{G}$  be its underlying Abelian group, and let  $\mathcal{A}$  be an idempotent reduct of  $\mathcal{M}$ . Then  $\mathcal{A}$  is a reduct of the algebra of  $\mathbf{E}_{\mathcal{G},r}$  for every  $r \geq 1$ .*

Bringing together these results with those of Section 4 we get:

**Theorem 15.** *Let  $\mathbf{B}$  be a finite structure and let  $\mathcal{B}$  be its algebra. If  $\text{var}(\mathcal{B})$  admits the unary or affine types, then there exists a non-trivial finite Abelian group  $\mathcal{G}$  such that  $\text{CSP}(\mathbf{E}_{\mathcal{G},r}) \leq_{\text{datalog}} \text{CSP}(\mathbf{B})$  for every  $r \geq 1$ .*

*Proof.* Since  $\text{CSP}(\mathbf{B}) = \text{CSP}(\text{core}(\mathbf{B}))$ , where  $\text{core}(\mathbf{B})$  is the core of  $\mathbf{B}$ , we may assume that  $\mathbf{B}$  is a core. Let  $\mathbf{D}$  be the singleton-expansion of  $\mathbf{B}$  and let  $\mathcal{D}$  be its algebra, which is idempotent. By Lemma 12, we have  $\text{CSP}(\mathbf{D}) \leq_{\text{datalog}} \text{CSP}(\mathbf{B})$ . Moreover, if  $\text{var}(\mathcal{B})$  admits types **1** or **2**, so does  $\text{var}(\mathcal{D})$  because  $\mathcal{D}$  is a reduct of  $\mathcal{B}$  (see [10, Chapter 5]). By Lemma 13, the variety  $\text{var}(\mathcal{D})$  contains a finite idempotent reduct  $\mathcal{A}$  of a module. Let  $\mathcal{G}$  be the Abelian group underlying the module. Then  $\mathcal{G}$  is non-trivial and finite. Moreover,  $\mathcal{A}$  is a reduct of the algebra of  $\mathbf{E}_{\mathcal{G},r}$  for every  $r \geq 1$  by Lemma 14. It follows that  $\text{CSP}(\mathbf{E}_{\mathcal{G},r}) \leq_{\text{datalog}} \text{CSP}(\mathbf{D})$ . Composing we get the result.  $\square$

We have seen in Section 3 that  $\text{CSP}(\mathbf{E}_{\mathcal{G},3})$  is not definable in  $C_{\infty\omega}^\omega$  when  $\mathcal{G}$  is non-trivial. Since definability in  $C_{\infty\omega}^\omega$  is preserved downwards by Datalog-reductions, this yields the following corollary:

**Corollary 16.** *Let  $\mathbf{B}$  be a finite structure and let  $\mathcal{B}$  be its algebra. If  $\text{CSP}(\mathbf{B})$  is definable in  $C_{\infty\omega}^\omega$ , then  $\text{var}(\mathcal{B})$  omits the unary and affine types.*

Corollary 16 can be seen as a strengthening of the result of Larose and Zádori [11] that if the complement of  $\text{CSP}(\mathbf{B})$  is definable in Datalog then  $\text{var}(\mathcal{B})$  omits the unary and affine types. Larose and Zádori also conjectured the converse, namely that if  $\text{var}(\mathcal{B})$  omits the unary and affine types then the complement of  $\text{CSP}(\mathbf{B})$

is definable in Datalog. By Corollary 16 this conjecture would imply that every  $\text{CSP}(\mathbf{B})$  is either definable in Datalog or not definable in  $C_{\infty\omega}^{\omega}$ , which can be seen as a definability dichotomy.

Finally, we consider three decision problems. In ALGEBRA OF TYPE 2 we are given a finite set  $A$  and operation tables of idempotent operations  $f_1, \dots, f_n$  on  $A$ , and the question is whether  $\text{var}(\mathcal{A})$ , where  $\mathcal{A} = (A; \{f_1, \dots, f_n\})$ , omits types 1 and 2. In RELATIONAL STRUCTURE OF TYPE 2 we are given a finite relational structure  $\mathbf{A}$ , and the question is whether  $\text{var}(\text{Al}(\mathbf{A}))$  omits types 1 and 2. In RELATIONAL STRUCTURE OF TYPE 2( $k$ ) we are given a finite relational structure  $\mathbf{A}$ ,  $|A| \leq k$ , and the question is whether  $\text{var}(\text{Al}(\mathbf{A}))$  omits types 1 and 2. The problems ALGEBRA OF TYPE 2 and RELATIONAL STRUCTURE OF TYPE 2( $k$ ) were shown tractable in [12].

**Theorem 17.** RELATIONAL STRUCTURE OF TYPE 2 is NP-complete.

## References

1. Feder, T., Vardi, M.: Computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing* 28, 57–104 (1998)
2. Dalmau, V.: Linear datalog and bounded path duality of relational structures. *Logical Methods in Computer Science* 1(1) (2005)
3. Larose, B., Loten, C., Tardif, C.: A characterisation of first-order constraint satisfaction problems. In: *LICS*, pp. 201–210 (2006)
4. Bulatov, A.: Mal'tsev constraints are tractable. Technical Report PRG-RR-02-05, Computing Laboratory, University of Oxford, Oxford, UK (2002)
5. Bulatov, A.A., Dalmau, V.: A simple algorithm for Mal'tsev constraints. *SIAM J. Comput.* 36(1), 16–27 (2006)
6. Dalmau, V., Kolaitis, P., Vardi, M.: Constraint satisfaction, bounded treewidth, and finite variable logics. In: Van Hentenryck, P. (ed.) *CP 2002*. LNCS, vol. 2470, pp. 311–326. Springer, Heidelberg (2002)
7. Kolaitis, P., Vardi, M.: A game-theoretic approach to constraint satisfaction. In: *Proc. 17th National Conference on Artificial Intelligence, AAAI-2000*, pp. 175–181 (2000)
8. Blass, A., Gurevich, Y., Shelah, S.: On polynomial time computation over unordered structures. *J. Symbolic Logic* 67(3), 1093–1125 (2002)
9. Bulatov, A., Jeavons, P., Krokhin, A.: Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing* 34, 720–742 (2005)
10. Hobby, D., McKenzie, R.: *The Structure of Finite Algebras*. Volume 76 of *Contemporary Mathematics*. American Mathematical Society, Providence, R.I. (1988)
11. Larose, B., Zádori, L.: Bounded width problems and algebras. *Algebra Universalis* (to appear)
12. Larose, B., Valeriote, M.: personal communication (2006)
13. Diestel, R.: *Graph Theory*. Springer, Heidelberg (1997)
14. Seymour, P., Thomas, R.: Graph searching and a min-max theorem for treewidth. *Journal of Combinatorial Theory Series B* 58, 22–33 (1993)
15. Libkin, L.: *Elements of Finite Model Theory*. Springer, Heidelberg (2004)

16. Ebbinghaus, H.D., Flum, J.: Finite Model Theory, 2nd edn. Springer, Heidelberg (1999)
17. Immerman, N.: Descriptive Complexity. Springer, Heidelberg (1999)
18. Hella, L.: Logical hierarchies in PTIME. *Info. and Comput.* 129, 1–19 (1996)
19. Burris, S., Sankappanavar, H.: A course in universal algebra. In: *Graduate Texts in Mathematics*, vol. 78, Springer, New York (1981)

# Boundedness of Monadic FO over Acyclic Structures

Stephan Kreutzer<sup>1</sup>, Martin Otto<sup>2</sup>, and Nicole Schweikardt<sup>1,\*</sup>

<sup>1</sup> Institut für Informatik, Humboldt Universität zu Berlin  
{kreutzer, schweika}@informatik.hu-berlin.de  
<sup>2</sup> Fachbereich Mathematik, Technische Universität Darmstadt  
otto@mathematik.tu-darmstadt.de

**Abstract.** We study the boundedness problem for monadic least fixed points as a decision problem. While this problem is known to be undecidable in general and even for syntactically very restricted classes of underlying first-order formulae, we here obtain a decidability result for the boundedness issue for monadic fixed points over arbitrary first-order formulae in restriction to acyclic structures.

## 1 Introduction

The extension of first-order logic by least and greatest fixed points of monotone first-order operators is one of the most natural remedies to some of the obvious limitations of first-order logic when considered as a query language over relational structures. While for instance the very basic graph query concerning reachability of a red node is well known not to be first-order expressible, it possesses a straightforward formalisation in terms of the monadic least fixed point associated with the positive, monotone first-order operator  $X \mapsto \{x: \text{red}(x) \vee \exists y(Exy \wedge Xy)\}$ . In a database context, the query language DATALOG is the extension of positive existential first-order logic by least fixed points. In the context of modal logics, the modal  $\mu$ -calculus extends basic modal logic by monadic least and greatest fixed points and provides one of the most prominent frameworks for temporal and process logics. In the context of descriptive complexity and finite model theory, the extension of FO by least and greatest fixed points plays a major role. In all these cases it is natural to ask which queries rely on fixed point recursion in an essential way — as opposed to queries which, although they are presented as fixed points, could also be formalised without.

Least fixed points of monotone operators are reached in a (transfinite) iteration of the operator starting from the empty set and taking unions at limit ordinals. The resulting monotone sequence of stages eventually reaches the fixed point. A fixed point process is *bounded* if there is a finite bound on the number of iterations required, uniformly across all input structures. As a decision problem, the *boundedness problem* asks, given a monotone relational operator, whether the least fixed point process for this operator is bounded. As a logical decision problem for first-order formulae, more specifically, we want to decide whether the monotone operator specified by some first-order formula  $\varphi(X, x)$  that is positive in  $X$  is bounded.

---

\* We gratefully acknowledge our participation in the 2006 Isaac Newton Institute programme on Logic and Algorithms; this opportunity has greatly promoted our collaboration in this research.

The boundedness problem was first studied with a view to query optimisation, in particular for variants of DATALOG. Clearly a bounded fixed point can be eliminated in favour of an explicit unfolding of the iteration to the required depth. By a classical theorem of Barwise and Moschovakis [3], cf. Theorem 2.1 below, boundedness of the fixed point process associated with  $\varphi(X, x) \in \text{FO}$  not just implies but is actually equivalent to first-order definability of the fixed point. This reduces the question whether a given fixed point can be eliminated in favour of any “static” first-order definition to the boundedness issue.

The model theoretic link between the procedural behaviour of fixed point recursions and FO-ness provides a source of interest in the boundedness problem that goes far beyond the original motivation from query optimisation. The study of effective criteria for FO-definability of MSO-definable properties has a long tradition in particular over word and tree structures, cf. the handbook chapter [17]. We recall the characterisations by McNaughton–Pappert and Schützenberger, and Beauquier–Pin, respectively, in the word case; and the more recent work of Benedikt and Segoufin [4] for trees.

Not surprisingly, the boundedness problem as such is undecidable. In fact one can show that the boundedness problem is at least as hard as the satisfiability problem (for fragments of FO satisfying some very mild closure conditions). But also for many fragments of FO that are decidable for satisfiability, like purely existential or purely universal FO or two-variable first-order logic  $\text{FO}^2$ , boundedness is known to be undecidable [10, 11, 13, 16]. On the positive side, boundedness is decidable for instance for monadic DATALOG [7] (purely existential positive FO) as well as for its universal counterpart [16], and for the modal fragment of FO [15]. But the decidability region is very narrow; for instance binary DATALOG and monadic DATALOG with inequality are undecidable [10, 11], and also monadic universal FO with equality or with mixed polarities in the static relations [14, 16]. The boundedness problem may thus be viewed as a critical strengthening of the satisfiability issue for fragments of FO; in fact unboundedness of  $\varphi$  precisely corresponds to the satisfiability of the sequence of formulae that express that the  $n$ -th stage of  $\varphi$  is non-trivial, for each  $n$ .

The undecidability proofs for the boundedness problem of partly very weak fragments of FO suggest that grid-like structures (or structures of unbounded treewidth) play a crucial role in the reduction arguments (in [13, 14, 16], in particular, tiling problems are used). This would explain, for instance, why binary recursion almost inevitably leads to undecidability: here the recursion itself can be used to generate grid-like structures into which other undecidable issues can be coded. For *monadic* recursion, on the other hand, grid-like structure can only reside in the input and not be built up in the fixed point process. It therefore seems conceivable that boundedness of monadic fixed points over certain classes of *tree-like* structures, e.g., structures of bounded treewidth, could be decidable. Such a decidability result, we hope, would provide a uniform framework for most of the known decidability results for monadic boundedness — just as reductions of the tiling problem can provide a rather uniform view of the negative cases, explored in [14].

We therefore want to analyse the boundedness issue from a new perspective, in a way orthogonal to the established approach: rather than looking at fragments of first-order logic, we ask whether decidability can be regained for *all* monadic first-order

fixed points *over suitably restricted classes of structures*. This can be viewed as being part of recent efforts towards developing a model theory for “well-behaved” classes of structures [2,11,15].

As a first significant result in this direction concerning boundedness issues, we here show the decidability of boundedness of monadic first-order fixed points over the class of *acyclic structures*, i.e., structures whose (undirected) Gaifman graph is acyclic (we consider such tree-like structures rather than, say, directed acyclic graphs, since boundedness is undecidable over the class of directed acyclic graphs: the two-dimensional grids that can be used to reduce the tiling problem to monadic boundedness issues can clearly be cast as directed acyclic graphs in this sense). A major goal for the extension of the present paper’s approach would consist in a similar decidability result for boundedness of monadic fixed points over classes of bounded treewidth.

## 2 Preliminaries

We denote first-order structures by German letters  $\mathfrak{A}, \mathfrak{B}, \dots$  and their universes by corresponding Roman letters  $A, B, \dots$ . We use  $\mathcal{C}, \mathcal{T}, \dots$  for classes of structures. We always assume that classes of structures are closed under isomorphisms. If  $\mathfrak{A}$  is a structure,  $U \subseteq A$ , and  $\psi(X)$  is a formula with a free monadic second-order variable  $X$ , we write  $(\mathfrak{A}, U) \models \psi$ , or  $\mathfrak{A} \models \psi[U]$ , whichever is more convenient, to denote that the expansion of  $\mathfrak{A}$  by  $X^{\mathfrak{A}} := U$  satisfies  $\psi$ . If  $\vartheta(X, x)$  is a formula with a free monadic second-order variable  $X$  and a free first-order variable  $x$ , we let  $\vartheta^{\mathfrak{A}}(U) := \{a \in A : \mathfrak{A} \models \vartheta[U, a]\}$ . We omit the index if  $\mathfrak{A}$  is understood.

Let  $\varphi(X, x)$  be a first-order formula with a free first-order variable  $x$  and a free second-order variable  $X$ , in which it is positive (i.e.,  $X$  only occurs within the scope of an even number of negation symbols). On corresponding structures  $\mathfrak{A}$ ,  $\varphi$  defines an operator

$$F_{\varphi}^{\mathfrak{A}} : \mathcal{P}(A) \longrightarrow \mathcal{P}(A) \quad \text{with} \quad F_{\varphi}^{\mathfrak{A}}(P) = \varphi^{\mathfrak{A}}(P) \quad \text{for each } P \in \mathcal{P}(A),$$

where  $\mathcal{P}(A)$  denotes the power set of  $A$ . Due to positivity in  $X$ ,  $F_{\varphi}^{\mathfrak{A}}$  is monotone and hence has a least fixed point  $(\mu X. \varphi)^{\mathfrak{A}}$  which we usually write as  $\varphi^{\infty}(\mathfrak{A})$ . The fixed point can also be obtained as the limit of the monotone sequence of stages  $X^{\alpha}$ , with  $\alpha$  an ordinal, defined by  $X^0 := \emptyset$ ,  $X^{\alpha+1} := \varphi^{\mathfrak{A}}(X^{\alpha})$ ,  $X^{\lambda} := \bigcup_{\beta < \lambda} X^{\beta}$  for limit ordinals  $\lambda$ . We usually write  $\varphi^{\alpha}(\mathfrak{A})$  to denote the  $\alpha$ -th stage  $X^{\alpha}$ .

Note that each finite stage  $X^n$ ,  $n \in \mathbb{N}$ , is uniformly first-order definable. We write  $\varphi^n(x)$  for the formula that defines the  $n$ -th stage of  $\varphi$ , which is obtained inductively by substituting  $\varphi^{n-1}(x)$  for each atom  $Xx$  in  $\varphi(X, x)$ , where  $\varphi^0(x)$  is meant to be false.

$F_{\varphi}$  (or the least fixed point of  $\varphi$  or also just  $\varphi$ ) is called *bounded* (over a class of structures  $\mathcal{C}$ ) if there is some  $n \in \mathbb{N}$  such that  $\varphi^{\infty}(\mathfrak{A}) = \varphi^n(\mathfrak{A})$  for all  $\mathfrak{A}$  (for all  $\mathfrak{A} \in \mathcal{C}$ ).

Recall that a class of structures is *elementary* if it is the class of models of some first-order theory; a class is called *projective* if it is the class of models of some first-order theory in a possibly extended vocabulary (cf., e.g., [12]). The notions of FO-*definability* (and similarly, *projective FO-definability*, as well as MSO-*definability*) refer to definability in terms of single sentences rather than possibly infinite theories.

At various places throughout the paper we will use the following classical theorem [3] on boundedness.

**Theorem 2.1 (Barwise-Moschovakis).** *The following are equivalent for every FO formula  $\varphi(X, x)$  suitable for positive least fixed-point iteration (also in restriction to any elementary or projective class of structures):*

- (i)  $\varphi(X, x)$  is bounded.
- (ii)  $\varphi^\infty$  is uniformly FO-definable.
- (iii)  $\varphi^\infty$  is FO-definable in each structure (non-uniformly).

We also remark that the Löwenheim–Skolem theorem for FO tells us that a first-order fixed point is bounded (over some elementary or projective class) if, and only if, it is bounded over all countable structures (in that class).

Recall that the *Gaifman graph*  $\mathcal{G}(\mathfrak{A})$  of a first-order structure  $\mathfrak{A} := (A, \tau^{\mathfrak{A}})$  of signature  $\tau$  is defined as the undirected graph with vertex set  $A$  and an edge between two vertices  $a, b \in A$ , if  $a \neq b$  and there exists an  $R \in \tau$  and a tuple  $(a_1, \dots, a_r) \in R^{\mathfrak{A}}$  such that  $a, b \in \{a_1, \dots, a_r\}$ .

**Definition 2.1.** *A structure is called acyclic if its Gaifman graph is acyclic.  $\mathcal{AC}$  denotes the class of all acyclic structures.*

Note that  $\mathcal{AC}$  is elementary so that the Barwise-Moschovakis theorem applies to it. For the rest of this paper we work over a fixed finite relational signature of unary and binary relation symbols. The restriction to at most binary signatures is w.l.o.g., as in this paper we only work with acyclic structures and a structure containing a relation  $R$  of arity  $> 2$  can only be acyclic if every tuple in  $R$  contains at most two distinct elements. Such relations can easily be coded in binary relations.

### 3 Locality

#### 3.1 Syntactic Locality and a Positive Variant of Gaifman’s Theorem

In 1981, Gaifman [9] proved that any first-order formula is equivalent to a Boolean combination of basic-local sentences and local formulae. We recall the necessary definitions.

Let  $\mathfrak{A} := (A, \tau^{\mathfrak{A}})$  be a first-order structure of signature  $\tau$ . The *distance*  $d^{\mathfrak{A}}(a, b)$  between two elements  $a, b \in A$  is defined as the length of the shortest path in the Gaifman graph  $\mathcal{G}(\mathfrak{A})$  connecting  $a$  and  $b$ . For  $r \geq 0$  and  $a \in A$  we define the *r-neighbourhood* of  $a$  in  $\mathfrak{A}$  as  $N_r^{\mathfrak{A}}(a) := \{b \in A : d^{\mathfrak{A}}(a, b) \leq r\}$ . It is easily seen that for any  $r \geq 0$  there is a first-order formula  $\delta_r(x, y) \in \text{FO}[\tau]$  such that  $\mathfrak{A} \models \delta_r[a, b]$  iff  $d^{\mathfrak{A}}(a, b) \leq r$ , for all  $\tau$ -structures  $\mathfrak{A}$  and all  $a, b \in A$ . For notational convenience we write  $d(x, y) \leq r$  for  $\delta_r(x, y)$  and  $d(x, y) > r$  for  $\neg\delta_r(x, y)$ .

If  $\varphi(x)$  is a first-order formula, then  $\varphi^{N_r(x)}(x)$  is the formula obtained from  $\varphi$  by relativising all quantifiers in  $\varphi$  to the  $r$ -neighbourhood of  $x$ , i.e. replacing  $\forall y\psi$  by  $\forall y(d(x, y) \leq r \rightarrow \psi)$  and  $\exists y\psi$  by  $\exists y(d(x, y) \leq r \wedge \psi)$ . A formula  $\psi(x)$  of the form  $\varphi^{N_r(x)}(x)$  is called *r-local*. A formula  $\psi(x)$  is *local*, or *local in x*, if it is  $r$ -local for some  $r \geq 0$ .

**Theorem 3.1 (Gaifman [9]).** *Every first-order formula  $\varphi(x)$  is equivalent to a Boolean combination of local formulae  $\chi(x)$ , and basic local sentences, i.e., sentences of the form*

$$\exists x_1 \dots \exists x_k \left( \bigwedge_{1 \leq i < j \leq k} d(x_i, x_j) > 2r \wedge \bigwedge_{1 \leq i \leq k} \vartheta(x_i) \right)$$

for suitable  $r, k > 0$  and an  $r$ -local formula  $\vartheta(x)$ .

The following theorem establishes a variant of Gaifman's locality theorem for first-order formulae  $\varphi(X, x)$  which are positive in the monadic second-order variable  $X$ . The proof is an adaptation of the proof of the analogous statement for sentences as it appears in [8]. Due to space limitations, we refrain from giving the proof here and refer the reader to the full version of the paper.

**Theorem 3.2.** *Every formula  $\varphi(X, x) \in \text{FO}$  that is positive in the monadic second-order variable  $X$  is logically equivalent to a finite disjunction of formulae  $\varphi_i(X, x) \wedge \psi_i(X)$ , where the  $\varphi_i(X, x)$  are local in  $x$  and the  $\psi_i$  are positive in  $X$  and conjunctions of (possibly negated) basic local sentences. Furthermore, for every formula  $\varphi(X, x) \in \text{FO}$  that is positive in  $X$ , we can effectively compute a finite disjunction of formulae  $\varphi_i(X, x) \wedge \psi_i(X)$  that is equivalent to  $\varphi$  over  $\mathcal{AC}$ .*

In what follows we shall actually not even rely on the basic local nature of the  $X$ -positive sentential components  $\psi_i(X)$ .

Note that the  $\varphi_i(X, x)$  are local in  $x$  but not necessarily positive in  $X$ . The following example demonstrates that the theorem fails if in addition we require the  $\varphi_i(X, x)$  to be positive in  $X$ . Let  $\tau := \{E, P, X\}$ , where  $E$  is binary and  $P$  and  $X$  are unary, and consider the formula  $\varphi(X, x) := Px \wedge \exists y(x \neq y \wedge Xy \wedge Py)$ . Suppose that  $\varphi(X, x)$  is equivalent to a formula  $\psi := \bigvee_{i=1}^k (\varphi_i(X, x) \wedge \psi_i(X))$ , where the  $\varphi_i$  are positive in  $X$  and local in  $x$  and the  $\psi_i$  are positive in  $X$ . Let  $\mathfrak{A} := (\{a, b\}, \tau^{\mathfrak{A}})$  with  $E^{\mathfrak{A}} := \emptyset$  and  $P^{\mathfrak{A}} := \{a, b\}$  and  $X^{\mathfrak{A}} := \{b\}$ . Clearly,  $\mathfrak{A} \models \varphi[a]$  and therefore there exists an  $i \in \{1, \dots, k\}$  such that  $\mathfrak{A} \models (\varphi_i \wedge \psi_i)[a]$ . The  $l$ -neighbourhoods of  $a$  and  $b$  are distinct for all  $l$  and their  $\{E, P\}$ -reducts are isomorphic. As  $\mathfrak{A} \models \varphi_i[a]$  and  $\varphi_i$  is positive in  $X$  it follows that  $\mathfrak{A} \models \varphi_i[b]$ . Hence,  $\mathfrak{A} \models (\varphi_i \wedge \psi_i)[b]$  but  $\mathfrak{A} \not\models \varphi[b]$  contradicting the equivalence of  $\varphi$  and  $\psi$ .

### 3.2 Locality of Queries

**Definition 3.3** *Let  $\mathcal{C}$  be a class of  $\tau$ -structures.*

- (i) *A monadic query  $Q$  on  $\mathcal{C}$  is a mapping which assigns to each  $\mathfrak{A} \in \mathcal{C}$  a set  $Q^{\mathfrak{A}} \subseteq A$  so that for all isomorphisms  $\pi : \mathfrak{A} \cong \mathfrak{B}$  and all  $a \in A, a \in Q^{\mathfrak{A}} \iff \pi(a) \in Q^{\mathfrak{B}}$ .*
- (ii) *A monadic query  $Q$  is MSO-definable, if there is an MSO-formula  $\varphi(x) \in \text{MSO}[\tau]$  such that  $Q^{\mathfrak{A}} = \{a \in A : \mathfrak{A} \models \varphi[a]\}$  for all  $\mathfrak{A} \in \mathcal{C}$ .*
- (iii) *Let  $l \in \mathbb{N}$ .  $Q$  is  $l$ -local over  $\mathcal{C}$ , if for all countable structures  $\mathfrak{A}, \mathfrak{B} \in \mathcal{C}$  and  $a \in A, b \in B$  such that  $(\mathfrak{A}|_{N_l^{\mathfrak{A}}(a)}, a) \cong (\mathfrak{B}|_{N_l^{\mathfrak{B}}(b)}, b)$ :  $a \in Q^{\mathfrak{A}} \iff b \in Q^{\mathfrak{B}}$ .*

Here,  $\mathfrak{A}|_{N_l^{\mathfrak{A}}(a)}$  denotes the restriction of  $\mathfrak{A}$  to the  $l$ -neighbourhood of  $a$ .

- (iv)  *$Q$  is local, if it is  $l$ -local for some  $l \in \mathbb{N}$ .*



Note that locality for queries is a purely semantic property, as opposed to syntactic locality of formulae discussed above. We refer to just *countable* structures in (iii), as this will allow us to use interpretations of the relevant structures in the  $\omega$ -branching tree; for our boundedness concerns we may always restrict attention to countable structures, by the Löwenheim–Skolem theorem (cf., Section 2).

**Lemma 3.4.** *Let  $\varphi(x) \in \text{MSO}$  and let  $Q$  be the query defined by  $\varphi$  over a class  $\mathcal{C} \subseteq \mathcal{AC}$ . If  $Q$  is  $r$ -local for some  $r \in \mathbb{N}$ , then  $Q$  is FO-definable over  $\mathcal{C}$ , in fact even by a local first-order formula.*

*In other words, a local query  $Q$  is MSO-definable if, and only if, it is FO-definable.*

*Proof (Sketch).* The proof is obtained on the basis of a decomposition argument for MSO-types and the observation that MSO-formulas of quantifier rank  $q$  can count multiplicities of types only up to a threshold  $g(q)$ . This can be used to translate  $r$ -local MSO-formulae into FO-formulae by induction on the locality radius  $r$ .  $\square$

## 4 Boundedness over Acyclic Structures

The goal of this section is to prove the following theorem, the main result of the paper. Recall Definition 2.1 for the class of acyclic structures  $\mathcal{AC}$ .

**Theorem 4.1.** *Let  $\mathcal{C} \subseteq \mathcal{AC}$  be an FO-definable subclass of  $\mathcal{AC}$  and let  $\varphi(X, x)$  be a first-order formula positive in the monadic second-variable  $X$ . Then boundedness of  $\varphi$  in  $\mathcal{C}$  is decidable.*

The main ingredients for the proof are the following. In Section 4.2, we use Theorem 3.2 to show that deciding boundedness for arbitrary first-order formulae  $\varphi(X, x)$ , positive in  $X$ , can be decomposed into a sequence of boundedness tests for purely local formulae  $\varphi_i(X, x)$  in certain projectively FO-definable subclasses  $\mathcal{C} \subseteq \mathcal{AC}$ .

To decide boundedness for local formulae  $\varphi_i(X, x)$ , we show that this can further be reduced to deciding whether the global relation  $\varphi^\infty(x)$  defined by  $\varphi_i(X, x)$  is local. Locality of MSO-definable queries in MSO-definable subclasses of  $\mathcal{AC}$  can be decided via reduction to the MSO-theory of the  $\omega$ -branching tree and Rabin’s theorem, see Section 4.1.

### 4.1 Locality Testing for MSO-Queries

**Theorem 4.2.** *Let  $\mathcal{C} \subseteq \mathcal{AC}$  be an MSO-definable subclass of  $\mathcal{AC}$  and let  $Q$  be a query that is MSO-definable over  $\mathcal{C}$ . Then locality of  $Q$  over  $\mathcal{C}$  is decidable. The decision procedure is uniform in the MSO-formula defining the subclass  $\mathcal{C}$ .*

The remainder of this section is devoted to the proof of the theorem, which is based on a reduction to Rabin’s following classical result.

**Theorem 4.3 (Rabin).** *The MSO-theory of the infinite  $\omega$ -branching tree is decidable.*

We denote the infinite  $\omega$ -branching tree as  $\mathfrak{T}_\omega$ , its root as  $\lambda$ . The first step towards the proof of Theorem 4.2 is to show that there is a uniform MSO-interpretation that interprets structures  $\mathfrak{A} \in \mathcal{C}$  in colourings of  $\mathfrak{T}_\omega$  in a suitable way. We can then use the decidability of the MSO-theory of  $\mathfrak{T}_\omega$  to check if the query is local. This method follows ideas from [15][16] and in particular allows us to capture locality (boundedness) in the framework of MSO over trees through a regular analogue of König’s lemma that is available in  $\mathfrak{T}_\omega$ .

Let  $\mathcal{C}$  be an MSO-definable subclass of  $\mathcal{AC}$  over a signature  $\sigma$ . Given a  $\sigma$ -structure  $\mathfrak{A} \in \mathcal{AC}$  and an element  $a \in A$  we encode  $\mathfrak{A}, a$  in a colouring of  $\mathfrak{T}_\omega$  by a suitable set  $\tau := \tau(\sigma)$  of colours: every tree in the Gaifman graph  $\mathcal{G}(\mathfrak{A})$  is coded in a subtree rooted at a successor of the root  $\lambda$  so that the distinguished element  $a$  is encoded by a direct successor of  $\lambda$  and marked by a special colour. We refrain from giving details here as the encoding is straightforward. It is important, however, that the encoding preserves the distance between elements of the structure within each component of  $\mathcal{G}(\mathfrak{A})$ .

The interpretation allows us to translate MSO-formulae over structures from  $\mathcal{AC}$  to MSO-formulae over encodings in  $\mathfrak{T}_\omega$ . In particular, there is an MSO-sentence  $\psi$  that is true in a  $\tau$ -expansion  $\mathfrak{T}$  of  $\mathfrak{T}_\omega$  if, and only if,  $\mathfrak{T}$  encodes some structure  $\mathfrak{A} \in \mathcal{C}$ .

Let  $\varphi(x)$  be an MSO-formula defining a query  $Q$  over  $\mathcal{C}$  (which again can be translated to a corresponding formula  $\varphi'(x)$  over  $\tau$ -expansions of  $\mathfrak{T}_\omega$ ). For an acyclic structure  $\mathfrak{A}$  with distinguished element  $a \in A$  and  $r \geq 1$  we let  $\mathfrak{A}|_r := \mathfrak{A}|_{N_r^{\mathfrak{A}}(a)}$  denote the initial segment of all nodes up to depth  $r$ . A set  $A' \subseteq A$  is *initial*, if it contains  $a$  and is connected (in the Gaifman graph).  $A'$  is called *local*, if  $A' \subseteq \mathfrak{A}|_r$  for some  $r$ . It is *path-finite* if it contains no infinite paths. Note that while it is MSO-definable that a subset of  $\mathcal{T}_\omega$  is initial or path-finite, locality is not (this follows from König’s lemma together with the fact that every MSO-formula that is satisfiable in  $\mathfrak{T}_\omega$  is also satisfiable in a finitely branching tree).

A *regular* expansion of  $\mathfrak{T}_\omega$  is one that realises only finitely many isomorphism types of subtrees. The following regular analogue of König’s lemma is proved in [16].

**Lemma 4.4.** *An initial subset  $D$  in a regular expansion  $(\mathfrak{T}_\omega, D)$  of  $\mathfrak{T}_\omega$  is path-finite if, and only if, it is local.*

Let  $\varphi_{local}(Z)$  be an MSO-formula that says of an initial subset  $Z$  of  $\mathfrak{T}_\omega$  rooted at some immediate successor  $z$  of the root  $\lambda$ , that whenever encodings of two structures  $\mathfrak{A}, \mathfrak{B} \in \mathcal{C}$  in  $\mathfrak{T}_\omega$ , whose distinguished elements  $a \in A$  and  $b \in B$  are represented by  $z$ , agree on  $Z$ , then  $\mathfrak{A} \models \varphi[a]$  if, and only if,  $\mathfrak{B} \models \varphi[b]$ . Clearly, this is MSO-definable. Analogous to the reasoning in [16], we can show the following lemma.

**Lemma 4.5.** *Let  $\lambda$  be the root of  $\mathfrak{T}_\omega$ . The following are equivalent:*

- (i)  $\varphi(x)$  is local
- (ii) for some  $r \in \mathbb{N}$  and for  $D = \mathfrak{T}_\omega|_r$ :  $\mathfrak{T}_\omega \models \varphi_{local}[D]$ .
- (iii)  $\mathfrak{T}_\omega \models \exists Z (Z \text{ is initial and path-finite} \wedge \varphi_{local}(Z))$ .
- (iv) there is a regular expansion  $(\mathfrak{T}_\omega, D)$  of  $\mathfrak{T}_\omega$  with an initial, path-finite  $D$  such that  $\mathfrak{T}_\omega \models \varphi_{local}[D]$ .

*Proof.* (i)  $\implies$  (ii)  $\implies$  (iii) are obvious. (iii)  $\implies$  (iv) is a well-known fact about MSO. For (iv)  $\implies$  (i) use Lemma 4.4. □

By Rabin's theorem, part (iii) of the previous lemma, and hence locality of  $\varphi$ , is decidable. This completes the proof of Theorem 4.2.

## 4.2 Boundedness of Arbitrary Monadic FO-formulae

We show that the boundedness problem for arbitrary monadic first-order formulae over acyclic structures can be reduced to the locality test for MSO-formulae as provided by Theorem 4.2.

Let  $\varphi(X, x) \in \text{FO}$  be a formula, positive in  $X$ . By Theorem 3.2,  $\varphi$  is equivalent to  $\bigvee_i (\varphi_i(X, x) \wedge \psi_i(X))$ , where the  $\varphi_i(X, x)$  are local in  $x$  and the  $\psi_i$  are positive in  $X$ . To simplify the presentation, we only consider the case where

$$\varphi \equiv (\varphi_1(X, x) \wedge \psi_1(X)) \vee (\varphi_2(X, x) \wedge \psi_2(X)).$$

We can treat the  $\psi_i$  as *guards*, that enable or disable the contribution of  $\varphi_i$  to the fixed-point induction, in the sense that certain  $\psi_i$  may be true already at the initial stage (and stay true for the whole induction process) whereas others may only become true at later stages. In this case, the corresponding  $\varphi_i$  only contribute to the induction process beginning with these stages and can be neglected before.

Inductions over  $\varphi$  can therefore be decomposed into a sequence of *phases*. Let  $\mathfrak{A}$  be a structure. We distinguish between several cases:

- (0)  $\mathfrak{A} \not\models \psi_1[\emptyset]$  and  $\mathfrak{A} \not\models \psi_2[\emptyset]$ . Then  $\varphi^\infty(\mathfrak{A}) = \emptyset$ , so we can ignore this case.
- (1)  $\mathfrak{A} \models \psi_1[\emptyset]$  and  $\mathfrak{A} \models \psi_2[\emptyset]$ . In this case, the induction on  $\varphi$  in  $\mathfrak{A}$  is equivalent to the induction on  $(\varphi_1 \vee \varphi_2)$ , which is purely local.
- (2)  $\mathfrak{A} \models \psi_1[\emptyset]$  and  $\mathfrak{A} \not\models \psi_2[\varphi^\infty(\mathfrak{A})]$ . In this case, the induction on  $\varphi$  in  $\mathfrak{A}$  is equivalent to the induction on  $\varphi_1$ , as the guard  $\psi_2$  for  $\varphi_2$  will never become true.
- (3)  $\mathfrak{A} \models \psi_1[\emptyset]$  and  $\mathfrak{A} \models \psi_2[\varphi^\infty(\mathfrak{A})]$ . Over  $\mathfrak{A}$ , the induction has two phases. It starts with an induction on  $\varphi_1$ . As soon as  $\psi_2(X^\alpha)$  is satisfied for a stage  $\alpha$ , the induction continues on  $\varphi_1 \vee \varphi_2$ .

There are two more cases, just like (2) and (3) but with the roles of  $\varphi_1/\psi_1$  and  $\varphi_2/\psi_2$  exchanged (we suppress this duplication in the following).

To illustrate the possible interplay between the various phases, let us consider formulae in positive Gaifman form of the kind

$$\varphi(X, x) = \psi_0 \wedge [Px \vee (\varphi_1(X, x) \wedge \exists x(Xx \wedge Q_1x)) \vee (\varphi_2(X, x) \wedge \exists x(Xx \wedge Q_2x))],$$

with local formulae  $\varphi_i(X, x)$ , and with extra unary predicates  $P$  and  $Q_i$ , which serve to initialise the fixed point process and to trigger phases driven by  $\varphi_1$ ,  $\varphi_2$  or their combination  $\varphi_1 \vee \varphi_2$ . Static side conditions in  $\psi_0$  and the local formulae  $\varphi_i$  may be such that, for instance, the induction starts on  $\varphi_1$  (e.g.,  $\psi_0$  may force  $Q_2 \cap P = \emptyset$ ), and that the induction on  $\varphi_1$  alone would be unbounded while the induction on  $\varphi_2$  alone would be bounded, but such that the unbounded induction on  $\varphi_1$  always triggers the induction on  $\varphi_2$  (due to  $Q_2$ -elements in the 17-th stage of  $\varphi_1$  in any model of  $\psi_0$  in which  $\varphi_1$  is unbounded, say). The interplay between  $\varphi_1$  and  $\varphi_2$  could now still be such that the overall process is bounded or unbounded. It should be clear from this naive analysis of a very simple family of examples that the full variety of phase patterns needs to be taken

into account and that several boundedness issues for purely local processes determine the overall boundedness in a non-trivial manner.

Let us proceed with the proof of Theorem 4.1. Considering the possible sources for unboundedness of  $\varphi$ , it is clear that  $\varphi$  is unbounded if, and only if, at least one of the following applies:

- Case 1:**  $\varphi_1 \vee \varphi_2$  is unbounded over  $\mathcal{C}_1 := \{\mathfrak{A} \in \mathcal{AC} : \mathfrak{A} \models (\psi_1 \wedge \psi_2)[\emptyset]\}$ .  
**Case 2:**  $\varphi_1$  is unbounded over  $\mathcal{C}_2 := \{\mathfrak{A} \in \mathcal{AC} : \mathfrak{A} \models \psi_1[\emptyset] \text{ and } \mathfrak{A} \not\models \psi_2[\varphi^\infty(\mathfrak{A})]\}$ .  
**Case 3:** a)  $\varphi_1 \vee \psi_2$  is unbounded over  $\mathcal{C}_3 := \{\mathfrak{A} \in \mathcal{AC} : \mathfrak{A} \models \psi_1[\emptyset]\}$ , or  
 b)  $\varphi_1 \vee \psi_2$  is bounded over  $\mathcal{C}_3$ , with some bound  $n \in \mathbb{N}$ , and  $\varphi_1 \vee \varphi_2$  is unbounded over  $\mathcal{C}_4 := \{\mathfrak{A} \in \mathcal{AC} : \mathfrak{A} \models \psi_1[\emptyset] \text{ and } \mathfrak{A} \models \psi_2[\varphi^n(\mathfrak{A})]\}$ .  
**Case 4:** like 2 and 3 with the roles of  $\varphi_1/\psi_1$  and  $\varphi_2/\psi_2$  exchanged.

We further reduce the number of the cases that need to be considered.

- Lemma 4.6.** (1)  $\varphi_1$  is unbounded over  $\mathcal{C}_2$  if, and only if,  $\varphi_1$  is unbounded over  $\mathcal{C}'_2 := \{\mathfrak{A} \in \mathcal{AC} : (\mathfrak{A}, \emptyset) \models \psi_1 \text{ and } (\mathfrak{A}, P) \not\models \psi_2 \text{ for some fixed point } P \text{ of } \varphi_1\}$ .  
 (2) If  $\varphi_1 \vee \psi_2$  is unbounded over  $\mathcal{C}_3$ , then  $\varphi_1$  is unbounded over  $\mathcal{C}_2$ .  
 (3) If  $\varphi_1 \vee \psi_2$  is bounded over  $\mathcal{C}_3$ , then  $\varphi$  is unbounded over  $\mathcal{C}_3$  if, and only if,  $\varphi' := \varphi_1 \vee \varphi_2 \vee Px$  is unbounded over  $\mathcal{C}'_3 := \{(\mathfrak{A}, P) : \mathfrak{A} \in \mathcal{AC}, \mathfrak{A} \models \psi_1[\emptyset], \mathfrak{A} \models \psi_2[P], P \subseteq \varphi^\infty[\mathfrak{A}]\}$ .

Note that  $\mathcal{C}'_2$  is projectively FO-definable, as the fact that  $P$  is a fixed point of  $\varphi_1$  is FO-definable with a new relation for  $P$ . Both  $\mathcal{C}'_3$  and  $\mathcal{C}'_2$  are clearly MSO-definable.

*Proof.* (1) is trivial. For (2), assume  $\varphi_1 \vee \psi_2$  is unbounded over  $\mathcal{C}_3$  but  $\varphi_1$  is bounded over  $\mathcal{C}_2$ . Consider the theories  $T_0 := \{\exists x(\varphi^{n+1}(x) \wedge \neg\varphi^n(x)) : n \in \mathbb{N}\}$  and

$$T := T_0 \cup \{\neg\psi_2(\varphi^n) : n \in \mathbb{N}\} \cup (\text{axiomatisation of } \mathcal{C}_3).$$

$T$  is consistent, as  $\varphi_1 \vee \psi_2$  is unbounded over  $\mathcal{C}_3$ . Note that  $T$  speaks about  $\varphi$ , not  $\varphi_1$ , but  $\varphi^n \equiv \varphi_1^n$  in models of  $T$  as  $T$  implies  $\psi_1$  and  $\neg\psi_2(\varphi^n)$  for all  $n \in \mathbb{N}$ . By assumption,  $\varphi$  is bounded over  $\mathcal{C}_2$  and hence  $T_0$  is not satisfiable in  $\mathcal{C}_2$ . Therefore,

$$\text{every model of } T \text{ satisfies } \psi_2(\varphi^\infty), \quad (*)$$

as  $\neg\psi_2(\varphi^\infty)$  is the defining condition of the subclass  $\mathcal{C}_2$  of  $\mathcal{C}_3$ . But in every model  $\mathfrak{A}$  of  $\psi_2(\varphi^\infty)$ ,  $(\varphi \vee \psi_2)^\infty(\mathfrak{A}) = (\varphi_1 \vee \psi_2)^\infty(\mathfrak{A}) = A$  and with (4.2), therefore,  $(\varphi \vee \psi_2)^\infty(\mathfrak{A}) = (\varphi_1 \vee \psi_2)^\infty(\mathfrak{A}) = A$  in every model  $\mathfrak{A}$  of  $T$ . Furthermore, the equivalence between the first and the second induction is even stage-by-stage.

Hence, in the model class of  $T$ , the fixed point  $(\varphi \vee \psi_2)^\infty$  is trivially first-order definable – it is defined by *true*. Therefore, by the Barwise-Moschovakis Theorem,  $(\varphi \vee \psi_2)$  is bounded in models of  $T$ , contradicting  $T_0$ .

(3) Note that the phase in the generation of  $\varphi^\infty$  over some  $\mathfrak{A} \in \mathcal{C}_3$  that is driven by  $\varphi_1 \vee \varphi_2$  is stage-by-stage equivalent to the fixed point generation of  $(\varphi')^\infty$  over  $(\mathfrak{A}, \varphi^n(\mathfrak{A})) \in \mathcal{C}'_3$ , for the minimal  $n$  such that  $\mathfrak{A} \models \psi_2[\varphi^n(\mathfrak{A})]$ . Therefore boundedness of  $\varphi$  over  $\mathcal{C}_3$  trivially implies boundedness of  $\varphi'$  over  $\mathcal{C}'_3$ . Conversely, by decomposition into corresponding phases, boundedness of  $\varphi_1 \vee \psi_2$  and of  $\varphi'$  over  $\mathcal{C}'_3$  together imply that  $\varphi$  is bounded over  $\mathcal{C}_3$ .  $\square$

The desired decision procedure for boundedness of  $\varphi$  is built on a sequence of applications of decision procedures that detect (un)boundedness in the form of (non)locality. Consider the individual boundedness issues in cases 1–3 above in the light of their reformulation in Lemma 4.6:

- Case 1 corresponds to a boundedness test for a purely local fixed point:  $\varphi_1 \vee \varphi_2$  is purely local, and stage-by-stage equivalent with  $\varphi$  over the FO-definable subclass  $C_1$  of  $\mathcal{A}$ . As  $C_1$  is elementary, by the Barwise-Moschovakis Theorem, boundedness is equivalent to FO-definability of the fixed point. Further, as a bounded fixed point over a purely local formula is local itself, boundedness of  $\varphi_1 \vee \varphi_2$  is equivalent to *local* FO-definability. Finally, as  $C_1$  and the fixed point are MSO-definable, local FO-definability of the fixed point is decidable according to Theorem 4.2 and Lemma 3.4.
- Case 2 is similarly reduced to a boundedness test for a purely local fixed point over the projective and MSO-definable class  $C'_2$  in part 1 of the lemma.
- Case 3a) is in fact subsumed by case 2, according to part 2 of the lemma.

For what remains (case 3b) we may restrict attention to the situation where  $\varphi_1 \vee \psi_2$  is bounded over  $C_3$ . The corresponding decision issue obtained in part 3 of Lemma 4.6 is further reduced to a locality issue that can be decided according to Theorem 4.2 through condition (iii) in the following.

**Claim 4.7.** *Let  $\varphi_1 \vee \psi_2$  be bounded over  $C_3$ . Then the following are equivalent:*

- (i)  $\varphi$  is bounded over  $C_3$ .
- (ii)  $\varphi'$  is bounded over  $C'_3$ .
- (iii)  $(\varphi')^\infty$  is local over  $C'_3$ .

*Proof.* The equivalence between (i) and (ii) was dealt with in part 3 of Lemma 4.6. As  $\varphi'$  is purely local, boundedness clearly implies locality for the fixed point, i.e., (ii)  $\Rightarrow$  (iii). Assume now (iii). Obviously,  $(\varphi')^\infty$  is MSO-definable and therefore FO-definable due to locality, by Lemma 3.4. As  $\varphi_1 \vee \psi_2$  is bounded over  $C_3$ , there is some  $n$  such that for  $\mathfrak{A} \in C_3$ ,  $\mathfrak{A} \models \psi_2[\varphi^\infty(\mathfrak{A})] \iff \mathfrak{A} \models \psi_2[\varphi^n(\mathfrak{A})]$ . Further,  $\varphi'$  is stage-by-stage equivalent to  $\varphi \vee Px$  over  $C'_3$  and hence  $(\varphi \vee Px)^\infty$  is FO-definable over the elementary subclass  $C''_3 := \{\langle \mathfrak{A}, P \rangle : \mathfrak{A} \in \mathcal{A}, \mathfrak{A} \models \psi_1[\emptyset], \mathfrak{A} \models \psi_2[P], P = \varphi^n(\mathfrak{A})\} \subseteq C'_3$ . By the Barwise-Moschovakis theorem, therefore,  $\varphi \vee Px$  is bounded over  $C''_3$ , which implies that  $\varphi$  is bounded over  $C_3$ , since also  $\varphi_1 \vee \psi_2$  is bounded.  $\square$

It follows that testing for unboundedness of  $\varphi_1 \vee \varphi_2$  over  $C_1$ ;  
 else, for unboundedness of  $\varphi_1$  over  $C'_2$ ;  
 else, for unboundedness of  $\varphi'$  over  $C'_3$ ,

fails if, and only if,  $\varphi$  is bounded over  $\mathcal{A}$ . In the given format, each one of these unboundedness tests can be realised as a (non)locality test, which is effective through Theorem 4.2. This cascade of locality tests can be adapted to the (more complex) phase pattern of a formula in positive Gaifman form with more than two disjuncts. We also

<sup>1</sup> and the mirror symmetric case with  $\varphi_1/\psi_1$  exchanged for  $\varphi_2/\psi_2$ .

point out that all the arguments used relative to FO-definable subclasses of  $\mathcal{AC}$ . This concludes the proof of Theorem 4.1.  $\square$

Theorem 4.1 yields a decision procedure for the boundedness problem of FO on acyclic structures. However, the running time of the procedure grows non-elementarily with the formula size. There is no hope for an algorithm whose running time is bounded by an elementary function. As the first-order satisfiability problem over  $\mathcal{AC}$  is reducible to the boundedness problem over  $\mathcal{AC}$ , the non-elementary lower bound for satisfiability on  $\mathcal{AC}$  established in [6] gives us a non-elementary lower bound here.

## 5 Outlook

Given the decidability of the boundedness problem on acyclic structures, it is natural to ask whether, for instance, the result also holds for the class of trees or extends to the class of structures of treewidth at most  $k$ , for fixed  $k$ . We collect a few observations to indicate that the results and the methods used here are indeed rather sensitive to the underlying class of structures.

- Remark 5.1.** (a) *Boundedness over trees (connected acyclic structures) does not imply boundedness over all acyclic structures; similarly, boundedness over finite acyclic structures does not imply boundedness over all acyclic structures.*
- (b) *The Barwise-Moschovakis theorem is well known to fail over the class of all finite structures. It also fails over the class of finite acyclic structures, over the class of trees and over the class of finite trees.*
- (c) *The connection between locality and FO-definability of Lemma 3.4 fails over simple FO-definable classes of graphs of bounded treewidth.*

*Proof.* For (a) consider, over acyclic structures or over trees, a reachability query (in itself unbounded) modified by side a condition that renders the process trivial unless there are at least two nodes of degree 0 (impossible over trees), or unless there is precisely one vertex of degree 1 (impossible in *finite* acyclic structures).

For (b) consider over finite forests the fixed point process that evaluates to the well-founded part; while this process is unbounded, it evaluates to the full vertex set. Over (finite or infinite) trees, the unbounded fixed point based on  $\text{red}(x) \vee \exists y(Exy \wedge Xy) \vee \exists y(\text{root}(y) \wedge Xy)$  is in FO (equivalent to  $\exists x \text{red}(x)$ ).

For (c) consider connectivity over the FO-definable class of graphs that are disjoint unions of cycles and two-way infinite successor chains (treewidth 2). Clearly this query is in MSO but neither local nor FO. Joining all the nodes in such graphs to one new central vertex, one obtains an FO-definable class of graphs of treewidth 3, which is FO-bi-interpretable with the original class. As all structures in this class have diameter 2, any query is trivially local here.  $\square$

Clearly, these observations do not imply that boundedness is undecidable over corresponding classes. However, we need to develop new tools to show decidability. This is part of ongoing work.

As pointed out above, classes of bounded treewidth in particular provide an interesting wider framework for the analysis of the boundedness problem, because of its

relevance for other known decidable cases. Here, arguments based on compactness and the Barwise-Moschovakis theorem are still available, and so is in principle the phase analysis of Section 4.2. However, as indicated in (c) above, the direct link with locality is lost (at a more technical level, distances and locality are not preserved in the passage between the structures themselves and their tree representations). The search for alternative methods to decide FO-definability or boundedness in this context remains of particular interest.

## References

1. Atserias, A., Dawar, A., Grohe, M.: Preservation under extensions on well-behaved finite structures. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1437–1449. Springer, Heidelberg (2005)
2. Atserias, A., Dawar, A., Kolaitis, P.G.: On preservation under homomorphisms and unions of conjunctive queries. *Journal of the ACM* 53(2), 208–237 (2006)
3. Barwise, J., Moschovakis, Y.: Global inductive definability. *Journal of Symbolic Logic* 43, 521–534 (1978)
4. Benedikt, M., Segoufin, L.: Regular tree languages definable in FO and FO<sub>mod</sub>. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 327–339. Springer, Heidelberg (2005) Also see manuscript of journal paper
5. Benedikt, M., Segoufin, L.: Towards a characterization of order-invariant queries over tame structures. In: Ong, L. (ed.) CSL 2005. LNCS, vol. 3634, pp. 276–291. Springer, Heidelberg (2005)
6. Compton, K., Ward Henson, C.: A uniform method for proving lower bounds on the computational complexity of logical theories. In: Abramsky, S., Gabbay, D.M., Maibaum, T. (eds.) *Handbook of Logic in Computer Science. Logic and Algebraic Methods*, vol. 5, pp. 129–216. Oxford University Press, Oxford (2000)
7. Cosmadakis, S., Gaifman, H., Kanellakis, P., Vardi, M.: Decidable optimization problems for database logic programs. In: Proc. STOC’88, pp. 477–490 (1988)
8. Dawar, A., Grohe, M., Kreutzer, S., Schweikardt, N.: Approximation schemes for first-order definable optimisation problems. In: Proc. LICS’06, pp. 411–420 (2006)
9. Gaifman, H.: On local and non-local properties. In: Stern, J. (ed.) *Herbrand Symposium, Logic Colloquium ’81*, pp. 105–135. North-Holland, Amsterdam (1982)
10. Gaifman, H., Mairson, H., Sagiv, Y., Vardi, M.: Undecidable optimization problems for database logic problems. *Journal of the ACM* 40, 683–713 (1993)
11. Hillebrand, G., Kanellakis, P., Mairson, H., Vardi, M.: Undecidable boundedness problems for datalog programs. *Journal of Logic Programming* 25, 163–190 (1995)
12. Hodges, W.: *Model Theory*. Cambridge University Press, Cambridge (1993)
13. Kolaitis, P., Otto, M.: On the boundedness problem for two-variable first-order logic. In: Proc. LICS’98, pp. 513–524 (1998)
14. Kolaitis, P., Otto, M.: On the boundedness problem for fragments of first-order logic: undecidability results. Unpublished draft (1999)
15. Otto, M.: Eliminating recursion in the  $\mu$ -calculus. In: Meinel, C., Tison, S. (eds.) STACS 99. LNCS, vol. 1563, pp. 531–540. Springer, Heidelberg (1999)
16. Otto, M.: The boundedness problem for monadic universal first-order logic. In: Proc. LICS’06, pp. 37–46 (2006)
17. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.) *Handbook of Formal Language Theory*, vol. 3, pp. 389–455. Springer, Heidelberg (1997)

# Strong Price of Anarchy for Machine Load Balancing

Amos Fiat\*, Haim Kaplan\*, Meital Levy\*\*, and Svetlana Olonetsky\*

School of computer science, Tel Aviv University, Tel Aviv 69978, Israel  
{fiat,haimk,levymeit,olonetsk}@post.tau.ac.il

**Abstract.** As defined by Aumann in 1959, a strong equilibrium is a Nash equilibrium that is resilient to deviations by coalitions. We give tight bounds on the strong price of anarchy for load balancing on related machines. We also give tight bounds for  $k$ -strong equilibria, where the size of a deviating coalition is at most  $k$ .

**Keywords:** Game theory, Strong Nash equilibria, Load balancing, Price of Anarchy.

## 1 Introduction

Many concepts of game theory are now being studied in the context of computer science. This convergence of different disciplines raises new and interesting questions not previously studied in either of the original areas of study. Much of this interest in game theory within computer science is due to the seminal papers of Nisan and Ronen [17] and Koutsoupias and Papadimitriou [14].

A *Nash equilibrium* ([16]) is a state in a noncooperative game that is stable in the sense that no agent can gain from unilaterally switching strategies. There are many “solution concepts” used to study the behavior of selfish agents in a non-cooperative game. Many of these are variants and extensions of the original ideas of John Nash from 1951.

One immediate objection to Nash equilibria as a solution concept is that agents may in fact collude and jointly choose strategies so as to “profit”. There are many possible interpretations of the statement that a set of agents “profit” from collusion. One natural interpretation of this statement is the notion of a *strong equilibrium* due to Aumann [5], where no coalition of players have any joint deviation such that every member strictly benefits. Whereas mixed strategy Nash equilibria always exist for finite games [16], this is not in general true for strong equilibria.

---

\* Research partially supported by the Israel Science Foundation and the German Israel Foundation.

\*\* The research has been supported by the Eshkol Fellowship funded by the Israeli Ministry of Science.



Holzman and Law-Yone [13] characterized the set of congestion games that admit strong equilibria. The class of congestion games studied was extended by Rozenfeld and Tennenholtz in [18]. [18] also considered mixed strong equilibria and correlated mixed strong equilibria under various deviation assumptions, pure, mixed and correlated.

The term *price of anarchy* was coined by Koutsoupias and Papadimitriou [14]. This is the ratio between the cost of the worst-case Nash equilibria and the cost of the social optimum. A related notion is the price of stability defined in [3], the ratio between the cost of the best Nash equilibria and the cost of the social optimum. These concepts have been extensively studied in numerous settings, machine load balancing [14,15,10,7,9], network routing [19,6,8], network design [4,11,13,12], etc.

Andelman *et al.* [2] initiated the study of the *strong price of anarchy* (SPoA), the ratio of the worst case strong equilibria to the social optimum. The authors also define the notion of a  $k$ -strong equilibrium, where no coalition of size up to  $k$  has any joint deviation where all strictly benefit. Analogous definitions can be made for the  $k$ -strong price of anarchy.

One may argue that the strong price of anarchy (which is never worse than the price of anarchy) removes the element of poor coordination and is entirely due to selfishness. Likewise, the  $k$ -strong price of anarchy measures the cost of selfishness and restricted coordination (up to  $k$  agents at once).

Our work here is a direct continuation of the work of Andelman *et al.* [2], and addresses many of the open problems cited there, in particular in the context of a load balancing game. In this setting agents (jobs) choose a machine, and job  $j$  placed on machine  $i$  contributes  $w_j(i)$  to the load on machine  $i$ . Agents seek machines with small load, and the social cost usually considered is the makespan, *i.e.*, the maximal load on any machine. Whereas [2] considered strong price of anarchy and  $k$ -strong price of anarchy for unrelated machines, herein we primarily consider the strong price of anarchy for related machines (machines having an associated speed).

## Our Results

1. Czumaj and Vocking [10] showed that the price of anarchy for load balancing on related machine is  $\Theta(\log m / \log \log m)$ , we show that the strong price of anarchy for load balancing on related machine is  $\Theta(\log m / (\log \log m)^2)$ . This is our most technically challenging result.
2. We also give tight results for the problems considered by [2]:
  - (a) In [2] the strong price of anarchy for load balancing on  $m$  unrelated machines was shown to lie between  $m$  and  $2m - 1$ . We prove that the true value is always  $m$ .
  - (b) In [2], the  $k$ -strong price of anarchy for load balancing of  $n$  jobs on  $m$  unrelated machines is between  $O(nm^2/k)$  and  $\Omega(n/k)$ . We prove that the  $k$ -strong price of anarchy falls in between and is  $\Theta(m(n - m + 1)/(k - m + 1))$ .

Due to the space limit proofs of some lemma are omitted.

**Preliminaries.** A load balancing game consists of a set  $M = \{M_1, \dots, M_m\}$  of machines, a set  $N = \{1, \dots, n\}$  of jobs (agents). We use the terms machine  $i$  or machine  $M_i$  interchangeably. Each job  $j$  has a weight function  $w_j()$  such that  $w_j(i)$  is the running time of job  $j$  on machine  $M_i$ . When the machines are *unrelated* then  $w_j()$  is an arbitrary positive real function. For *related* machines, machine  $M_i$  has a speed  $v(i)$  and  $w_j(i) = w_j/v(i)$  where  $w_j$  is the weight of job  $j$ . In game theoretic terms, the set of strategies for job  $j$  is the set of machines  $M$ . A state  $S$  is an assignment of jobs to machines. Let  $m_S(j)$  be the machine chosen by job  $j$  in state  $S$ . The load on machine  $M_i$  in state  $S$  is  $\sum_{j|M_i=m_S(j)} w_j(i)$ .

Given a state  $S$  where job  $j$  is assigned to machine  $M_i$ , we say that the load observed by job  $j$  is the load on machine  $M_i$  in state  $S$ . The makespan of a state  $S$  is the maximum load of a machine in  $S$ . Jobs seek to minimize their observed load. The state OPT (the social optimum) is a state with minimal makespan. We also denote the makespan in state OPT by OPT, and the usage would be clear from the context.

A *strong equilibrium* is a state where no group of jobs can jointly choose an alternative set of strategies so that every job in the set has a reduced observed load in the new state. In a *k-strong equilibrium* we restrict such groups to include no more than  $k$  agents. The *strong price of anarchy* is the ratio between the makespan of the worst strong equilibrium and OPT. The *k-strong price of anarchy* is the ratio between the makespan of the worst  $k$ -strong equilibrium and OPT.

## 2 Related Machines

We assume that machines are indexed such that  $v(i) \geq v(j)$  for  $i < j$ . We first prove the lower bound and then the upper bound.

**Theorem 1.** *The strong price of anarchy for  $m$  related machines and  $n$  jobs is  $\Omega(\log m/(\log \log m)^2)$ .*

*Proof.* We partition the machines into  $\ell + 1$  groups. Let these groups be  $G_0, G_1, \dots, G_\ell$ . We further subdivide each group  $G_i, 0 \leq i < \ell$ , into  $\log \ell$  subgroups, where all machines within the same subgroup have the same speed, but machines from different subgroups of a group differ in speed. The group  $G_\ell$  consists of a single subgroup  $F_{\ell \log \ell}$ . In total, we have  $\ell \log \ell + 1$  subgroups  $F_0, F_1, \dots, F_{\ell \log \ell}$ , where subgroups  $F_0, \dots, F_{\log \ell - 1}$  are a partition of  $G_0, F_{\log \ell}, \dots, F_{2 \log \ell - 1}$  are the subgroups of  $G_1$ , etc.

Let  $m_j$  denote the number of machines in subgroup  $F_j, 0 \leq j \leq \ell \log \ell$ . Then  $m_0 = 1$ , and for subgroup  $F_{j+1}$  such that  $F_j \subset G_i$  we define  $m_{j+1} = (\ell - i) \times m_j$ , for  $0 \leq j < \ell \log \ell$ . It follows that the number of machines in subgroup  $F_{\ell \log \ell}$  is at least  $(\ell!)^{\log \ell}$  and therefore  $m \sim (\ell!)^{\log \ell}$  and  $\ell \sim \log m/(\log \log m)^2$ . We define the speed of all machines in subgroup  $F_j$  to be  $2^{(\ell \log \ell - j)}$ .

Consider the following state,  $S$ . Each machine of group  $G_i$  is assigned  $\ell - i$  jobs. Jobs that are assigned to machines in subgroup  $F_j$  have weight  $2^{(\ell \log \ell - j)}$ . As the speed of such machines is  $2^{(\ell \log \ell - j)}$ , it follows that each such job contributes

one to the load of the machine it is assigned to. *I.e.*, the load on all machines in  $G_i$  is  $\ell - i$ . Machines of  $F_{\ell \log \ell}$  have no jobs assigned to them.

The load on the machines in group  $G_0$  is  $\ell$  which is also the makespan in  $S$ . The minimal makespan (OPT) is attained by moving the jobs assigned to machines from  $F_j$  each to a separate machine of subgroup  $F_{j+1}$ , for  $0 \leq j < \ell \log \ell$ . The load on all machines is now  $2^{\ell \log \ell - j} / 2^{\ell \log \ell - (j+1)} = 2$ .

State  $S$  is a Nash equilibrium. A job assigned to a machine of subgroup  $F_j$  has no incentive to migrate to a machine with a lower indexed subgroup since the current load there is equal or higher to the current load it observes. There is no incentive to migrate to a higher indexed subgroup as it observes a load of at least  $j + 1 > j$ . We now argue that state  $S$  is not only a Nash Equilibrium but also a strong Nash equilibrium.

First, note that jobs residing on machines of group  $G_i$ ,  $0 \leq i \leq \ell - 2$ , have no incentive to migrate to machines of group  $G_j$ , for  $j \geq i + 2$ . This follows since the speed of each machine in group  $G_j$  is smaller by a factor of more than  $2^{\log \ell} = \ell$  from the speed of any machine in group  $G_i$ . Thus, even if the job is alone on such a machine, the resulting load is higher than the load currently observed by the job (current load is  $\leq \ell$ ). Thus, any deviating coalition has the property that jobs assigned to machines from group  $G_i$  may only migrate to machines from groups  $G_j$ , for  $j \leq i + 1$ .

Suppose that jobs that participate in a deviating coalition are from machines in groups  $G_i, G_{i+1}, \dots, G_j$ ,  $1 \leq i \leq j \leq \ell$ . The load on machines from group  $G_i$  holding participating jobs must strictly decrease since either jobs leave (and the load goes down) or jobs from higher or equal indexed groups join (and then the load must strictly go down too). If machines from group  $G_i$  have their load decrease, and all deviating jobs belong to groups  $i$  through  $j$ ,  $i < j$ , then there must be some machine  $M \in G_p$ ,  $i < p \leq j$ , with an increase in load. Jobs can migrate to machine  $M$  either from a machine in group  $G_{p-1}$ , or from a machine in group  $G_j$  for some  $j \geq p$ .

If a deviating job migrates from a machine in  $G_j$  for some  $j \geq p$  then this contradicts the increase in the load on  $M$ . The contradiction arises as such jobs will only join the coalition if they become strictly better off, and for this to happen the load on  $M$  should decrease.

However, this holds even if the deviating job migrates to  $M$  from a machine in  $G_{p-1}$ . The observed load for this job prior to deviating was  $\ell - (p - 1)$  and it must strictly decrease. A job that migrates to machine  $M$  from  $G_{p-1}$  increases the load by an integral value. A job that migrates away from machine  $M$  decreases the load by an integral value too. This implies that the new load on  $M$  must be an integer no larger than  $\ell - (p - 1)$ , which contradicts the increase in load on  $M$ .  $\square$

### 3 Upper Bound on Strong Price of Anarchy for Related Machines

We assume that the speeds of the machines are scaled so that OPT is 1. Let  $S$  be an arbitrary strong Nash equilibrium, and let  $\ell_{\max}$  be the maximum load of

a machine in  $S$ . Our goal is to give an upper bound on  $\ell_{\max}$ . When required, we may assume that  $\ell_{\max}$  is a sufficiently large constant. Recall that machines are ordered such that  $v(1) \geq v(2) \geq \dots \geq v(m) > 0$ . Let  $\ell(i)$  be the load on machine  $M_i$ , i.e., the total weight of jobs assigned to machine  $M_i$  is  $\ell(i)v(i)$ .

### 3.1 Sketch of the Proof

We prove that  $m = \Omega(\ell_{\max}^{\ell_{\max} \log \ell_{\max}})$ , which implies  $\ell_{\max} = O(\log m / (\log \log m)^2)$ . To show that  $m = \Omega(\ell_{\max}^{\ell_{\max} \log \ell_{\max}})$  we partition the machines into consecutive disjoint “phases” (Definition 1), with the property that the number of machines in phase  $i$  is  $\Omega(\ell)$  times the number of machines in phase  $i - 1$  (Lemma 1), where  $\ell$  is the minimal load in phases 1 through  $i$ .

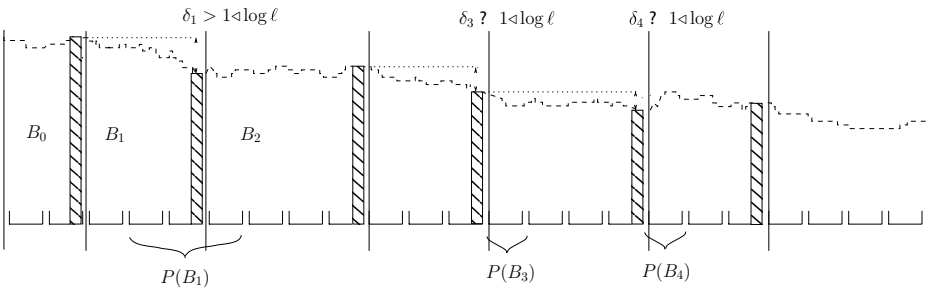
For technical reasons we introduce shifted phases (s-phases, Definition 2) which are in one-to-one correspondence to the phases. We focus on the s-phases of faster machines, so that the total drop in load amongst the machines of these s-phases is about  $\ell_{\max}/2$ . We next partition the s-phases into consecutive blocks. Let  $\delta_i$  be the load difference between slowest machine in block  $i - 1$  and the slowest machine in block  $i$ . By construction we get that  $\sum \delta_i = \Theta(\ell_{\max})$ .

We map s-phases to blocks such that each s-phase is mapped to at most one block as follows (Lemmas 4 and 5).

- If  $\delta_i < 1/\log \ell_{\max} \Rightarrow$  we map a single ( $1 = \lceil \delta_i \log \ell_{\max} \rceil$ ) s-phase to block  $i$
- If  $\delta_i \geq 1/\log \ell_{\max} \Rightarrow$  we map  $\Omega(\delta_i \log \ell_{\max})$  s-phases to block  $i$

Therefore the total number of s-phases is at least  $\sum \delta_i \log \ell_{\max} = \Omega(\ell_{\max} \log \ell_{\max})$ . Given the one-to-one mapping from s-phases to phases, this also gives us a lower bound of  $\Omega(\ell_{\max} \log \ell_{\max})$  on the number of phases.

Then we use Lemma 1 to conclude that the number of machines in phase  $i$  is  $\Omega(\ell_{\max})$  times the number of machines in phase  $i - 1$ . This allows us to conclude that the total number of machines  $m = \Omega(\ell_{\max}^{\ell_{\max} \log \ell_{\max}})$ , or that  $\ell_{\max} = O(\log m / (\log \log m)^2)$ .



**Fig. 1.** The machines are sorted in order of decreasing speed (and increasing index), and partitioned into s-phases. The s-phases are further partitioned into blocks  $B_i$ . The s-phases that are mapped to block  $i$  are marked  $P(B_i)$ .

### 3.2 Excess Weight and Excess Jobs

Given that the makespan of OPT is 1, the total weight of all jobs assigned to machine  $\sigma$  in OPT cannot exceed  $v(\sigma)$ , the speed of machine  $\sigma$ . We define the excess weight on machine  $1 \leq \sigma \leq m$  to be  $X(\sigma) = (\ell(\sigma) - 1)v(\sigma)$ . (Note that excess weight can be positive or negative).

Given a set  $R \subset \{1, \dots, m\}$ , we define the excess weight on  $R$  to be

$$X(R) = \sum_{\sigma \in R} X(\sigma). \tag{1}$$

For clarity of exposition, we use intuitive shorthand notation for sets  $R$  forming consecutive subsequences of  $1, \dots, m$ . In particular, we use the notation  $X(\sigma)$  for  $X(\{\sigma\})$ ,  $X(\leq w)$  for  $X(\{\sigma \mid 1 \leq \sigma \leq w\})$ ,  $X(w \dots y)$  for  $X(\{w \leq \sigma \leq y\})$ , etc.

Given that some set  $R$  has excess weight  $X(R) > 0$ , it follows that there must be some set of jobs  $J(R)$ , of total weight at least  $X(R)$ , that are assigned to machines in  $R$  by  $S$ , but are assigned to machines in  $\{1, \dots, m\} - R$  by OPT. Given sets of machines  $R$  and  $Q$ , let  $J(R \mapsto Q)$  be the set of jobs that are assigned by  $S$  to machines in  $R$  but assigned by OPT to machines in  $Q$ , and let  $X(R \mapsto Q)$  be the weight of the jobs in  $J(R \mapsto Q)$ . Let  $R, R_1$ , and  $R_2$  be a partition of the set of machines then we have

$$X(R) = X(R \mapsto \{1, \dots, m\} - R) = X(R \mapsto R_1) + X(R \mapsto R_2). \tag{2}$$

### 3.3 Partition into Phases

**Definition 1.** We partition the machines  $1, \dots, m$  into disjoint sets of consecutive machines called phases,  $\Phi_1, \Phi_2, \dots$ , where machines of  $\Phi_i$  precede those of  $\Phi_{i+1}$ . We define  $\rho_0 = 0$  and  $\rho_i = \max\{j \mid j \in \Phi_i\}$  for  $i \geq 1$ . Thus, it follows that  $\Phi_i = \{\rho_{i-1} + 1, \dots, \rho_i\}$ . It also follows that machines in  $\Phi_i$  are no slower than those of  $\Phi_{i+1}$ . Let  $n_i$  be number of machines in the  $i$ th phase, i.e.,  $n_i = \rho_i - \rho_{i-1}$ , for  $i \geq 1$ .

To determine  $\Phi_i$  it suffices to know  $\rho_{i-1}$  and  $\rho_i$ . For  $i = 1$  we define  $\rho_1 = 1$ , as  $\rho_0 = 0$  it follows that  $\Phi_1 = \{1\}$ . We define  $\rho_{i+1}$  inductively using both  $\rho_i$  and  $\rho_{i-1}$  as follows.

$$\rho_{i+1} = \operatorname{argmin}_{\sigma} \left\{ X(\leq \rho_i \mapsto \sigma) < X(\leq \rho_{i-1}) + \frac{X(\Phi_i)}{2} \right\}. \tag{3}$$

The phases have the following properties.

**Lemma 1.** Let  $\ell$  be the minimal load of a machine in phases  $1, \dots, i$ , ( $\ell = \min\{\ell(\sigma) \mid 1 \leq \sigma \leq \rho_i\}$ ), then  $n_{i+1} \geq n_i(\ell - 1)/2$ .

**Lemma 2.** For every  $i > j$ ,  $X(\Phi_i) > X(\Phi_j)$ .

Let  $\ell$  be the minimal load among machines  $1, \dots, \rho_i$ . Let  $\Gamma_i$  be the subset of  $\Phi_i$  that have at least  $(\ell - 1)/2$  of their load contributed by jobs of weight  $w \leq v(\rho_{i+1})$ . Let  $\bar{v}(\Gamma_i)$  be the average speed of the machines in  $\Gamma_i$ .

**Lemma 3.** For  $i > j$ ,  $\overline{v}(\Gamma_i)|\Gamma_i| \geq v(\rho_j)n_j(\ell - 1)/(\ell + 3)$ .

*Proof.* First we want to estimate  $X(\Phi_i \mapsto \geq \rho_{i+1})$ . By rewriting Equation (2) we get that

$$X(\Phi_i \mapsto \geq \rho_{i+1}) = X(\leq \rho_i \mapsto \geq \rho_{i+1}) - X(\leq \rho_{i-1} \mapsto \geq \rho_{i+1}) .$$

Since  $X(\leq \rho_{i-1} \mapsto \geq \rho_{i+1}) \leq X(\leq \rho_{i-1} \mapsto > \rho_i)$ , we also have that

$$X(\Phi_i \mapsto \geq \rho_{i+1}) \geq X(\leq \rho_i \mapsto \geq \rho_{i+1}) - X(\leq \rho_{i-1} \mapsto > \rho_i) . \tag{4}$$

From the definition of  $\rho_{i+1}$  follows that

$$X(\leq \rho_i \mapsto \geq \rho_{i+1}) \geq X(\leq \rho_{i-1}) + \frac{X(\Phi_i)}{2} , \tag{5}$$

Similarly, from the definition of  $\rho_i$  follows that

$$X(\leq \rho_{i-1} \mapsto > \rho_i) < X(\leq \rho_{i-2}) + \frac{X(\Phi_{i-1})}{2} . \tag{6}$$

Substituting Equations (5) and (6) into Equation (4) we get

$$\begin{aligned} X(\Phi_i \mapsto \geq \rho_{i+1}) &\geq X(\leq \rho_{i-1}) + \frac{X(\Phi_i)}{2} - \left( X(\leq \rho_{i-2}) + \frac{X(\Phi_{i-1})}{2} \right) \\ &\geq X(\leq \rho_{i-2}) + X(\Phi_{i-1}) + \frac{X(\Phi_i)}{2} - \left( X(\leq \rho_{i-2}) + \frac{X(\Phi_{i-1})}{2} \right) \\ &\geq \frac{X(\Phi_{i-1})}{2} + \frac{X(\Phi_i)}{2} . \end{aligned} \tag{7}$$

Let  $A(\sigma)$  be the total weight of jobs on machine  $\sigma$  with  $w(j) \leq v(\rho_{i+1})$  and let  $A(\Phi_i) = \sum_{\sigma \in \Phi_i} A(\sigma)$ . Since every job in  $J(\Phi_i \mapsto \geq \rho_{i+1})$  has weight of at most  $v(\rho_{i+1})$ , it follows that  $X(\Phi_i \mapsto \geq \rho_{i+1}) \leq A(\Phi_i)$ , and by Equation (7)

$$A(\Phi_i) \geq \frac{X(\Phi_{i-1})}{2} + \frac{X(\Phi_i)}{2} . \tag{8}$$

We claim that every machine  $\sigma$  with  $A(\sigma) > 0$  (i.e. the machine has at least one job  $j$  with  $w(j) \leq v(\rho_{i+1})$ ) has load of at most  $\ell + 1$ . To prove the claim, let  $q \leq \rho_{i+1}$  be a machine that has load greater than  $\ell + 1$  and a job  $j$  with  $w(j) \leq v(\rho_{i+1})$ , and let  $q'$  be the machine among  $1, \dots, \rho_i$  with load  $\ell$ . This state is not a Nash equilibrium since if job  $j$  switches to machine  $q'$  it would have a smaller cost. We get that

$$\begin{aligned} A(\Phi_i) &\leq \sum_{\sigma \in \Gamma_i} v(\sigma)(\ell + 1) + \sum_{\sigma \in \Phi_i - \Gamma_i} v(\sigma) \frac{\ell - 1}{2} \\ &= \sum_{\sigma \in \Gamma_i} v(\sigma) \frac{\ell + 3}{2} + \sum_{\sigma \in \Phi_i} v(\sigma) \frac{\ell - 1}{2} \\ &\leq \overline{v}(\Gamma_i)|\Gamma_i| \frac{\ell + 3}{2} + \frac{X(\Phi_i)}{2} . \end{aligned} \tag{9}$$

Inequality (9) holds since  $\bar{v}(\Gamma_i)$  is the average speed in  $\Gamma_i$  and  $\ell$  is the smallest load of a machine in  $1, \dots, \rho_i$ . Combining (9) and (8) we get that

$$\bar{v}(\Gamma_i)|\Gamma_i| \frac{\ell + 3}{2} + \frac{X(\Phi_i)}{2} \geq \frac{X(\Phi_{i-1})}{2} + \frac{X(\Phi_i)}{2}, \tag{10}$$

and therefore

$$\bar{v}(\Gamma_i)|\Gamma_i| \frac{\ell + 3}{2} \geq \frac{X(\Phi_{i-1})}{2} \geq \frac{X(\Phi_j)}{2} \geq v(\rho_j)n_j \frac{\ell - 1}{2}. \tag{11}$$

The first inequality in (11) follows from (10), the second follows from Lemma 2, and the third inequality follows since  $\ell$  is the smallest load of a machine in  $1, \dots, \rho_i$  and since  $v(\rho_j)$  is the smallest speed in  $\Phi_j$ . From (11) the lemma clearly follows.  $\square$

Recall that  $\ell_{\max}$  is the maximum load in  $S$ . Define  $k$  to be  $\min\{i|\ell(i) < \ell_{\max}/2\}$ . Let  $t$  be the phase such that  $\rho_t < k$  and  $\rho_{t+1} \geq k$ . Consider machines  $1, \dots, \rho_t$ . Let  $\ell$  be the minimal load of a machine in this set of machines. Then,  $\ell = \Theta(\ell_{\max})$ .

**Definition 2.** We define another partition of the machines into shifted phases ( $s$ -phases)  $\Psi_1, \Psi_2, \dots$  based on the partition to phases  $\Phi_1, \Phi_2, \dots$  as follows. We define  $\varphi_0 = 0$ . Let  $\varphi_i$  be the slowest machine in  $\Phi_i$  such that at least  $(\ell - 1)/2$  of its load is contributed by jobs with weight  $w \leq v(\rho_{i+1})$  (there exists such a machine by Lemma 3). We define  $\Psi_i = \{\varphi_{i-1} + 1, \dots, \varphi_i\}$ .

Note that there is a bijection between the  $s$ -phases  $\Psi_1, \Psi_2, \dots$  and the phases  $\Phi_1, \Phi_2, \dots$ . Furthermore, all machines in  $\Phi_i$  such that at least  $(\ell - 1)/2$  of their load is contributed by jobs of weight  $\leq v(\rho_{i+1})$  are in  $\Psi_i$ .

**Lemma 4.** The load difference between machines  $\varphi_2$  and  $\varphi_t$ ,  $\ell(\varphi_2) - \ell(\varphi_t) > \ell_{\max}/4 + 4$ .

We define  $z_{i+1}$  to be  $v(\varphi_i)/v(\varphi_{i+1})$ . Notice that  $z_{i+1} \geq 1$ . Let  $\Gamma(b) \subseteq \Psi_b$  be a subset of machines such that for every such machine, at least  $(\ell - 1)/2$  of the load is contributed by jobs with weight  $w \leq v(\varphi_{b+1})$ . For two  $s$ -phases  $\Psi_a$ , and  $\Psi_b$  the lemma below relates the difference in load of  $\varphi_a$  and  $\varphi_b$ , to the ratio of speeds  $v(\varphi_a)$  and  $v(\varphi_b)$ .

**Lemma 5.** Consider  $s$ -phases  $\Psi_a$  and  $\Psi_b$  such that  $a < b$ . Let  $\ell$  be the minimal load in  $\Psi_a$  and  $\Psi_b$ . If  $v(\varphi_a)/v(\varphi_b) \leq z_{a+1}(\ell - 1)/5$  then  $\ell(\varphi_a) \leq \ell(\varphi_b) + 4/z_{b+1}$ .

*Proof.* Proof by contradiction, assume that  $\ell(\varphi_a) = \ell(\varphi_b) + \alpha/z_{b+1}$ , for some  $\alpha > 4$ , and that  $v(\varphi_a)/v(\varphi_b) \leq z_{a+1}(\ell - 1)/5$ . We exhibit a deviating coalition all of whose members reduce their observed loads, contradicting the assumption that the current state is in strong equilibrium.

We observe that for every machine  $\sigma \in \Gamma(b)$  we have  $\ell(\sigma) \leq \ell(\varphi_b) + 1/z_{b+1}$ . If not, take any job  $j$  located on  $\sigma$ , such that  $w(\sigma) \leq v(\varphi_{b+1})$  and send it to machine  $\varphi_b$ , the contribution of job  $j$  to the load of  $\ell(\varphi_b)$  is at most  $v(\varphi_{b+1})/v(\varphi_b) =$

$1/z_{b+1}$ , i.e., the current state is not even a Nash equilibrium. Similarly, we have  $\ell(\varphi_b) \leq \ell(\sigma) + 1/z_{b+1}$ . (From this also follows that  $\ell(\sigma) < \ell(\varphi_a)$ .)

We group jobs on  $\varphi_a$  in a way such that the current load contribution of each group is greater than  $1/(2z_{a+1})$  and no more than  $1/z_{a+1}$ . I.e., for one such group of jobs  $G$ ,  $1/(2z_{a+1}) < \sum_{j \in G} w_j/v(\varphi_a) \leq 1/z_{a+1}$ . At least  $z_{a+1}(\ell - 1)/2$  such groups are formed. Every such group is assigned a unique machine in  $\Gamma(b)$  and all jobs comprising the group migrate to this machine. Let  $\Gamma \subseteq \Gamma(b)$  be a subset of machines that got an assignment,  $|\Gamma| = \min\{z_{a+1}(\ell - 1)/2, |\Gamma(b)|\}$ . The load contributed by migrating jobs to the target machine,  $\sigma \in \Gamma(b)$ , is therefore

$$\sum_{j \in G} \frac{w_j}{v(\sigma)} \leq \sum_{j \in G} \frac{w_j}{v(\varphi_b)},$$

we also know that  $v(\varphi_a)/v(\varphi_b) \leq z_{a+1}(\ell - 1)/5$  and  $\sum_{j \in G} w_j/v(\varphi_a) \leq 1/z_{a+1}$ , this gives us that

$$\sum_{j \in G} \frac{w_j}{v(\varphi_b)} \leq \sum_{j \in G} \frac{w_j}{v(\varphi_a)} \cdot \frac{v(\varphi_a)}{v(\varphi_b)} \leq (\ell - 1)/5.$$

Therefore, after migration, the load on  $\sigma \in \Gamma(b)$  is  $\leq \ell(\sigma) + (\ell - 1)/5 \leq \ell(\varphi_a) + (\ell - 1)/5$ . It is also at least  $\ell(\varphi_a)$  (otherwise  $S$  is not a Nash equilibrium).

Additionally, jobs will also migrate from machines  $\sigma \in \Gamma$  to machine  $\varphi_a$  (not the same jobs previously sent the other way). We choose jobs to migrate from  $\sigma \in \Gamma$  to  $\varphi_a$ , so that the final load on  $\sigma$  is strictly smaller than  $\ell(\varphi_a)$  and at least  $\ell(\varphi_a) - 1/z_{b+1} = \ell(\varphi_b) + (\alpha - 1)/z_{b+1}$ . It has to be smaller than  $\ell(\varphi_a)$  to guarantee that every job migrating from  $\varphi_a$  to  $\sigma$  observes a load strictly smaller than the load it observed before the deviation. We want it to be at least  $\ell(\varphi_b) + (\alpha - 1)/z_{b+1}$ , so that a job migrating to  $\varphi_a$  from  $\sigma$  would observe a smaller load as we will show below. To achieve this, slightly more than  $(\ell - 1)/5$  of the load of  $\sigma \in \Gamma$  has to migrate back to  $\varphi_a$ .

The jobs that migrate from  $\sigma \in \Gamma$  to  $\varphi_a$  are those jobs with load  $\leq 1/z_{b+1}$  on  $\sigma$ . Therefore, each such job which leaves  $\sigma$  reduces the load of  $\sigma$  by at most  $1/z_{b+1}$ . Since the total load of these jobs on  $\sigma$  is  $(\ell - 1)/2 > (\ell - 1)/5$ , we can successively send jobs from  $\sigma$  to  $\varphi_a$  until the load drops below to some value  $y$  such that  $\ell(\varphi_b) + (\alpha - 1)/z_{b+1} \leq y < \ell(\varphi_b) + \alpha/z_{b+1}$ .

We argued that prior to any migration, the load  $\ell(\sigma) \leq \ell(\varphi_b) + 1/z_{b+1}$  for  $\sigma \in \Gamma(b)$ . Following the migrations above, the new load  $\bar{\ell}(\sigma)$  on machine  $\sigma$  is  $\bar{\ell}(\sigma) \geq \ell(\varphi_b) + (\alpha - 1)/z_{b+1}$ . Thus, the load on every such machine has gone up by at least  $(\alpha - 2)/z_{b+1}$ .

If  $|\Gamma| = z_{a+1}(\ell - 1)/2$  the net decrease in load on machine  $\varphi_a$  is at least

$$\begin{aligned} \sum_{\sigma \in \Gamma} \frac{\alpha - 2}{z_{b+1}} \cdot \frac{v(\sigma)}{v(\varphi_a)} &\geq \sum_{\sigma \in \Gamma} \frac{\alpha - 2}{z_{b+1}} \cdot \frac{v(\varphi_b)}{v(\varphi_a)} \\ &\geq \frac{z_{a+1}(\ell - 1)}{2} \cdot \frac{\alpha - 2}{z_{b+1}} \cdot \frac{5}{z_{a+1}(\ell - 1)} \\ &\geq \frac{2.5(\alpha - 2)}{z_{b+1}} > \frac{\alpha + 1}{z_{b+1}}. \end{aligned}$$



If  $|\Gamma| < z_{a+1}(\ell - 1)/2$ , then  $\Gamma = \Gamma(b)$  and the net decrease in load on machine  $\varphi_a$  is at least

$$\begin{aligned} \sum_{\sigma \in \Gamma(b)} \frac{\alpha - 2}{z_{b+1}} \cdot \frac{v(\sigma)}{v(\varphi_a)} &\geq \frac{\alpha - 2}{z_{b+1}v(\varphi_a)} \sum_{\sigma \in \Gamma(b)} v(\sigma) \\ &\geq \frac{\alpha - 2}{z_{b+1}} \cdot \frac{\bar{v}(\Gamma(b))|\Gamma(b)|}{v(\varphi_a)} \\ &\geq \frac{2.5(\alpha - 2)}{z_{b+1}} > \frac{\alpha + 1}{z_{b+1}}. \end{aligned}$$

Thus, the new load  $\bar{\ell}(\varphi_a)$  on machine  $\varphi_a$  is at most

$$\bar{\ell}(\varphi_a) < \ell(\varphi_b) + \alpha/z_{b+1} - (\alpha + 1)/z_{b+1} = \ell(\varphi_b) - 1/z_{b+1},$$

which ensures that the jobs that migrate to machine  $\varphi_a$  could form a coalition, benefiting all members, in contradiction to the strong equilibrium assumption.  $\square$

We define a partition of the s-phases into blocks  $B_0, B_1, \dots$ . The first block  $B_0$  consists of the first two s-phases. Given blocks  $B_0, \dots, B_{j-1}$ , define  $B_j$  as follows: For all  $i$ , let  $a_i$  be the first s-phase of block  $B_i$  and let  $b_i$  be the last s-phase of block  $B_i$ . The first s-phase of  $B_j$  is s-phase  $b_{j-1} + 1$ , i.e.,  $a_j = b_{j-1} + 1$ .

To specify the last phase of  $B_j$  we define a consecutive set of s-phases denoted by  $P_1$ , where  $b_j \in P_1$ . The first s-phase in  $P_1$  is  $a_j$ . The last s-phase of  $P_1$  is the first phase, indexed  $p$ , following  $a_j$ , such that  $v(\varphi_{b_{j-1}})/v(\varphi_p) > z_{a_j}(\ell - 1)/5$ . Note that  $P_1$  always contains at least two s-phases. Let  $m_1$  be an s-phase in  $P_1 \setminus \{a_j\}$  such that  $z_{m_1} \geq z_i$  for every  $i$  in  $P_1 \setminus \{a_j\}$ . We consider two cases:

- Case 1:  $z_{m_1} \geq \log \ell$ . We define  $b_j = m_1 - 1$ . In this case we refer to  $B_j$  as a block of a type I.
- Case 2:  $z_{m_1} < \log \ell$ . We define  $P_2$  to be the suffix of  $P_1$  containing all s-phases  $i$  for which  $v(\varphi_{b_{j-1}})/v(\varphi_i) \geq z_{a_j}((\ell - 1)/5)^{2/3}$ . Note that s-phase  $p$  is in  $P_2$  and s-phase  $a_j$  is not in  $P_2$ . Let  $m_2$  be an s-phase in  $P_2$  such that  $z_{m_2} \geq z_i$  for every  $i$  in  $P_2$ . We define  $b_j$  to be  $m_2 - 1$ . In this case we refer to  $B_j$  as a block of type II.

If  $v(\varphi_{b_{j-1}})/v(\varphi_t) \leq z_{a_j}(\ell - 1)/5$  we do not define  $B_j$  and  $B_{j-1}$  is the last block.

For each block  $B_j$  let  $P(B_j)$  be the s-phases which we map to  $B_j$ . In Case 1 we define  $P(B_j) = m_1 = a_{j+1}$ . In Case 2 we define  $P(B_j) = P_2$ .

**Lemma 6.** *The number of phases associated with block  $B_j$ ,  $|P(B_j)|$ , is  $\Omega(\log \ell / z_{a_{j+1}})$ .*

*Proof.* If  $z_{m_1} \geq \log \ell$  then  $P(B_j)$  consists of a single phase. As  $\log \ell / z_{m_1} < 1$ , the claim trivially follows. Assume that  $z_{m_1} < \log \ell$ . Let  $s$  be the first s-phase in  $P_2$ , then

$$v(\varphi_{b_{j-1}})/v(\varphi_{s-1}) \leq z_{a_j} \left( \frac{\ell - 1}{5} \right)^{2/3}. \tag{12}$$

Let  $k$  be the last s-phase of  $P_2$  (which is also the last s-phase of  $P_1$ ), we have that

$$v(\varphi_{b_{j-1}})/v(\varphi_k) \geq z_{a_j} \frac{\ell - 1}{5} . \tag{13}$$

If we divide (13) by (12) we obtain that  $v(\varphi_k)/v(\varphi_{s-1}) \geq (\ell - 1/5)^{1/3}$ . Let  $q$  be the number of s-phases in  $P_2$ . Since  $z_{m_2} \geq z_i$  for all  $i \in P_2$  it follows that  $(z_{m_2})^q \geq (\frac{\ell-1}{5})^{1/3}$ . We conclude that  $q = \Omega(\log \ell / z_{m_2}) = \Omega(\log \ell / z_{a_{j+1}})$ , as  $\log x \leq x$  for all  $x$ .  $\square$

The following lemma shows each s-phase is mapped into at most one block.

**Lemma 7.** *For every pair of blocks  $B$ , and  $B'$  we have  $P(B) \cap P(B') = \emptyset$ .*

We now conclude the proof of the upper bound of the strong price of anarchy. By definition, we have that  $v(\varphi_{b_{j-1}})/v(\varphi_{b_j}) \leq z_{a_j}(\ell - 1)/5$ , so using Lemma 5 we get that

$$\ell(\varphi_{b_{j-1}}) - \ell(\varphi_{b_j}) \leq 4/z_{a_j} . \tag{14}$$

Let  $f$  be the index of the last block. We have that  $v(\varphi_{b_f})/v(\varphi_t) \leq z_{b_{f+1}}(\ell - 1)/5$ , (where  $t$  is the last phase with minimal load  $> \ell_{\max}/2$ ) so by Lemma 5,  $\ell(\varphi_{b_f}) \leq \ell(\varphi_t) + 4$ . By Lemma 4,  $\ell(\varphi_2) - \ell(\varphi_t) \geq \ell_{\max}/4 + 4$ . Therefore,  $\ell(\varphi_2) - \ell(\varphi_{b_f}) \geq \ell_{\max}/4$ . This together with Equation (14) gives that

$$\ell(\varphi_2) - \ell(\varphi_{b_f}) = \sum_{j=1, \dots, f} (\ell(\varphi_{b_{j-1}}) - \ell(\varphi_{b_j})) \leq \sum_{j=1, \dots, f} 4/z_{b_{j+1}} = \Theta(\ell_{\max}) . \tag{15}$$

Using Lemma 6 and Inequality (15) the total number of s-phases is

$$\sum_{i=1, \dots, f} \Omega(\log \ell) / z_{b_{i+1}} = \log \ell \sum_{i=1, \dots, f} 1/z_{b_{i+1}} = \Omega(\ell_{\max} \log \ell_{\max}) .$$

As described in the proof sketch this gives  $\ell_{\max} = O(\log m / (\log \log m)^2)$  as required. We conclude:

**Theorem 2.** *The strong price of anarchy for  $m$  related machines is*

$$\Theta(\log m / (\log \log m)^2) .$$

### 4 Unrelated Machines

**Strong Price of Anarchy.** We can show that the strong price of anarchy for  $m$  unrelated machine load balancing is at most  $m$ , improving the  $2m - 1$  upper bound given by Andelman *et al.* [2]. Our new upper bound is tight since it matches the lower bound shown in [2].

**Theorem 3.** *The strong price of anarchy for  $m$  unrelated machine load balancing is at most  $m$ .*

**$k$ -Strong Price of Anarchy.** In this section we consider coalitions of size at most  $k$ , where  $k \geq m$  (for  $k < m$  the upper bound is unbounded). Andelman et al. [2] show that for  $m$  machines and  $n \geq m$  players the  $k$ -strong price of anarchy is  $O(nm^2/k)$  and  $\Omega(n/k)$ . We give a refined analysis:

**Theorem 4.** *The  $k$ -strong price of anarchy for  $m$  unrelated machine load balancing,  $k \geq m$ , and given  $n$  jobs,  $c = \Theta(m(n-m)/(k-m))$ , more precisely,  $(m-1)(n-m+1)/(k-m+1) \leq c \leq 2m(n-m+1)/(k-m+2)$ .*

## References

1. Albers, S., Eilts, S., Even-Dar, E., Mansour, Y., Roditty, L.: On nash equilibria for a network creation game. In: SODA, pp. 89–98 (2006)
2. Andelman, N., Feldman, M., Mansour, Y.: Strong price of anarchy. In: SODA (2007)
3. Anshelevich, E., Dasgupta, A., Kleinberg, J., Tardos, E., Wexler, T., Roughgarden, T.: The price of stability for network design with fair cost allocation. In: FOCS, pp. 295–304 (2004)
4. Anshelevich, E., Dasgupta, A., Tardos, E., Wexler, T.: Near-optimal network design with selfish agents. In: STOC, pp. 511–520 (2003)
5. Aumann, R.J.: Acceptable points in general cooperative  $n$ -person games. In: Tucker, A.W., Luce, R.D. (eds.) Contribution to the Theory of Games. Annals of Mathematics Studies, 40, vol. IV, pp. 287–324 (1959)
6. Awerbuch, B., Azar, Y., Epstein, A.: Large the price of routing unsplitable flow. In: STOC, pp. 57–66 (2005)
7. Awerbuch, B., Azar, Y., Richter, Y., Tsur, D.: Tradeoffs in worst-case equilibria. In: Solis-Oba, R., Jansen, K. (eds.) WAOA 2003. LNCS, vol. 2909, pp. 41–52. Springer, Heidelberg (2003)
8. Christodoulou, G., Koutsoupias, E.: The price of anarchy of finite congestion games. In: STOC, pp. 67–73 (2005)
9. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination mechanisms. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 345–357. Springer, Heidelberg (2004)
10. Czumaj, A., Vöcking, B.: Tight bounds for worst-case equilibria. In: SODA, pp. 413–420 (2002)
11. Fabrikant, A., Luthra, A., Maneva, E., Papadimitriou, C., Shenker, S.: On a network creation game. In: PODC, pp. 347–351 (2003)
12. Fiat, A., Kaplan, H., Levy, M., Olonetsky, S., Shabo, R.: On the price of stability for designing undirected networks with fair cost allocations. In: ICALP, pp. 608–618 (2006)
13. Holzman, R., Law-Yone, N.: Strong equilibrium in congestion games. Games and Economic Behavior 21(1-2), 85–101 (1997)
14. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: STACS, pp. 404–413 (1999)
15. Mavronicolas, M., Spirakis, P.: The price of selfish routing. In: STOC, pp. 510–519 (2001)
16. Nash, J.: Non-cooperative games. Annals of Mathematics 54(2), 286–295 (1951)
17. Nisan, N., Ronen, A.: Algorithmic mechanism design (extended abstract). In: STOC (1999)
18. Rozenfeld, M.T.O.: Strong and correlated strong equilibria in monotone congestion games. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) WINE 2006. LNCS, vol. 4286, pp. 74–86. Springer, Heidelberg (2006)
19. Roughgarden, T., Tardos, E.: How bad is selfish routing? J. ACM 49(2), 236–259 (2002)

# Efficient Algorithms for Constant Well Supported Approximate Equilibria in Bimatrix Games\*

Spyros C. Kontogiannis<sup>1,2</sup> and Paul G. Spirakis<sup>2</sup>

<sup>1</sup> Department of Computer Science, University Campus, 45110 Ioannina, Greece

[kontog@cs.uoi.gr](mailto:kontog@cs.uoi.gr)

<sup>2</sup> Research Academic Computer Technology Institute, N. Kazantzaki Str., Patra  
University Campus, 26500 Rio-Patra, Greece

[{kontog,spirakis}@cti.gr](mailto:{kontog,spirakis}@cti.gr)

**Abstract.** In this work we study the tractability of well supported approximate Nash Equilibria (SuppNE in short) in bimatrix games. In view of the apparent intractability of constructing Nash Equilibria (NE in short) in polynomial time, even for bimatrix games, understanding the limitations of the approximability of the problem is of great importance.

We initially prove that SuppNE are immune to the addition of arbitrary real vectors to the rows (columns) of the row (column) player's payoff matrix. Consequently we propose a polynomial time algorithm (based on linear programming) that constructs a  $0.5$ -SuppNE for arbitrary win lose games.

We then parameterize our technique for win lose games, in order to apply it to arbitrary (normalized) bimatrix games. Indeed, this new technique leads to a *weaker*  $\phi$ -SuppNE for win lose games, where  $\phi = \frac{\sqrt{5}-1}{2}$  is the *golden ratio*. Nevertheless, this parameterized technique extends nicely to a technique for arbitrary  $[0, 1]$ -bimatrix games, which assures a  $0.658$ -SuppNE in polynomial time.

To our knowledge, these are the *first* polynomial time algorithms providing  $\varepsilon$ -SuppNE of normalized or win lose bimatrix games, for some nontrivial *constant*  $\varepsilon \in [0, 1)$ , bounded away from 1.

**Keywords:** Bimatrix Games, Well Supported Approximate Equilibria.

## 1 Introduction

One of the most appealing concepts in game theory is the notion of a Nash equilibrium: *A collection of strategies for the players from which no player has an incentive to unilaterally deviate.* The extremely nice thing about Nash equilibria is that they always exist in any finite  $k$ -person game in normal form [19]. This is one of the most important reasons why Nash equilibria are considered to be the prevailing solution concept for strategic games. The problem with Nash

---

\* This work was partially supported by the 6th Framework Programme under contract 001907 (DELIS).

equilibria is that there can be exponentially many of them, of quite different characteristics, even for bimatrix games. Additionally, we do not know yet how to construct them (not even approximately). Therefore,  $k$ -NASH, the problem of computing an arbitrary Nash equilibrium of a finite  $k$ -person game in normal form, is a fundamental problem in algorithmic game theory and perhaps one of the most outstanding problems at the boundary of  $\mathbf{P}$  [21]. Its complexity has been a long standing open problem, since the introduction of the pioneering pivoting algorithm of Lemke and Howson [17]. Unfortunately, it was recently shown by Savani and von Stengel [22] that this algorithm requires an exponential number of steps, and its smoothed complexity is also superpolynomial. It is also quite interesting that many (quite natural) refinements of  $k$ -NASH are known to be  $\mathbf{NP}$ -complete problems [13,7].

A flurry of results in the last two years has proved that  $k$ -NASH is indeed  $\mathbf{PPAD}$ -complete, even for four players [9], three players [12], and two players [4]. In particular, the result of Chen and Deng [4], complemented by that of Abbott, Kane and Valiant [1], shows that 2-NASH is  $\mathbf{PPAD}$ -complete even for win lose bimatrix games. Essentially, it is now clear that the complexity class  $\mathbf{PPAD}$ , introduced by Papadimitriou [20], indeed captures the whole class of finite 2-person games in normal form.

Due to the apparent hardness even of 2-NASH, approximate solutions to Nash equilibria have lately attracted the attention of the research community. There are two different notions of approximate Nash equilibria: Those which require that each player gets the maximum possible payoff, within some additive constant  $\varepsilon$  (denoted here by  $\text{ApproxNE}$ ), and those which require that each player is allowed to adopt  $\mathbf{wpp}$ <sup>1</sup> only actions that are *approximate pure best responses*, ie, at most an *additive* term  $\varepsilon$  smaller than the payoff of a best response strategy against the (fixed) strategy of the opponent (denoted here by  $\text{SuppNE}$ ).  $\text{ApproxNE}$  seem to be the dominant notion of approximate equilibria in the literature, while  $\text{SuppNE}$  is a rather new notion (eg, see [5,6,10,16]). As it will be explained later,  $\text{SuppNE}$  seem to be much harder notions of approximate equilibria to construct. On the other hand they might be naturally motivated by the players' selfish behavior: Rather than demanding that a player adopts  $\mathbf{wpp}$  only pure best responses against the opponent's strategy, we allow them to choose *approximate pure best responses*. This is in contrast to the notion of  $\text{ApproxNE}$ , in which the two players have no restriction in what kind of actions they choose to play  $\mathbf{wpp}$ , so long as their payoffs are close to their best response payoffs. We would like to argue in this paper that  $\text{SuppNE}$  is a quite interesting notion of approximate Nash equilibria, due to both its mathematical challenge and also its additional property that the players are not allowed to adopt  $\mathbf{wpp}$  actions that are indeed meaningless to them.

The present paper is a work providing (to our knowledge) the *first polynomial time algorithms* for  $\text{SuppNE}$  in arbitrary (normalized or win lose) bimatrix games, for some constant that is clearly *bounded away* from the trivial constant of 1. Indeed, for win lose games we provide an algorithm for 0.5- $\text{SuppNE}$ , which

---

<sup>1</sup> With positive probability.

is close to the strongest result of 0.38–ApproxNE for the much weaker notion of approximate NE, provided in [11]. For arbitrary (normalized) bimatrix games, we provide an algorithm for 0.658–SuppNE.

## 2 Preliminaries

### 2.1 Mathematical Notation

For any integer  $k \in \mathbb{N}$ ,  $[k] \equiv \{1, 2, \dots, k\}$ . We denote by  $M \in F^{m \times n}$  any  $m \times n$  matrix whose elements have values in some set  $F$ . We also denote by  $(A, B) \in (F \times F)^{m \times n}$  any  $m \times n$  matrix whose elements are pairs of values from  $F$ . Equivalently, this structure can be seen as a pair of  $m \times n$  matrices  $A, B \in F^{m \times n}$ . Such a matrix is called a **bimatrix** and we denote the  $m \times n$  matrix of the first coordinates of its elements by  $A$ , while the matrix of the second coordinates of its elements is  $B$ . A  $k \times 1$  matrix is also considered to be an  **$k$ -vector**. Vectors are denoted by bold small letters (eg,  $\mathbf{x}, \mathbf{y}$ ). A vector having a 1 in the  $i$ -th position and 0 everywhere else is denoted by  $\mathbf{e}_i$ . We denote by  $\mathbf{1}_k$  ( $\mathbf{0}_k$ ) the  $k$ -vector having 1s (0s) in all its coordinates. The  $k \times k$  matrix  $E = \mathbf{1}_k \cdot \mathbf{1}_k^T \in \{1\}^{k \times k}$  has value 1 in all its elements. For a pair of vectors  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ ,  $\mathbf{x} \geq \mathbf{y}$  denotes their *component-wise comparison*. Matrices are denoted by capital letters (eg,  $A, B, C, \dots$ ), and bimatrices are denoted by ordered pairs of capital letters (eg,  $(A, B), (R, C), \dots$ ). For any  $m \times n$  (bi)matrix  $M$ ,  $M_j$  is its  $j$ -th column (as an  $m \times 1$  vector),  $M^i$  is the  $i$ -th row (as a (transposed)  $1 \times n$  vector) and  $M_{i,j}$  is the  $(i, j)$ -th element.

For any integer  $r \geq 1$ , we denote by  $\Delta_r \equiv \{\mathbf{z} \in \mathbb{R}^r : \mathbf{z} \geq \mathbf{0}; (\mathbf{1}_r)^T \mathbf{z} = 1\}$  the  $(r - 1)$ -simplex. The subset  $\Delta_r(k) = \{\mathbf{x} \in \Delta_r : k\mathbf{x} \in \mathbb{Z}^r\}$  denotes the points in  $\Delta_r$  whose elements are multiples of  $\frac{1}{k}$ . Also,  $\hat{\Delta}_r(k) = \Delta_r \cap \{0, \frac{1}{k}\}^r$  contains the points in the  $(r - 1)$ -simplex that assign probability mass either zero or  $\frac{1}{k}$  to each element. For any point  $\mathbf{z} \in \Delta_r$ , its **support**  $\text{supp}(\mathbf{z})$  is the set of coordinates with positive value:  $\text{supp}(\mathbf{z}) \equiv \{i \in [r] : z_i > 0\}$ .

For an arbitrary logical expression  $\mathcal{E}$ , we denote by  $\mathbb{P}\{\mathcal{E}\}$  the probability of this expression being true, while  $\mathbb{I}_{\{\mathcal{E}\}}$  is the indicator variable of whether this expression is true or false. For any random variable  $x$ ,  $\mathbb{E}\{x\}$  is its expected value.

### 2.2 Game Theoretic Definitions and Notation

An  $m \times n$  **bimatrix game**  $\langle A, B \rangle$  is a 2-person game in normal form, that is determined by the bimatrix  $(A, B) \in (\mathbb{R} \times \mathbb{R})^{m \times n}$  as follows: The first player (called the **row player**) has an  $m$ -element *action set*  $[m]$ , and the second player (called the **column player**) has an  $n$ -element *action set*  $[n]$ . Each row (column) of the bimatrix corresponds to a different action of the row (column) player. The row and the column player's payoffs are determined by the  $m \times n$  real matrices  $A$  and  $B$  respectively. In the special case that the payoff matrices have only rational entries, we refer to a **rational bimatrix game**. If both payoff matrices belong to  $[0, 1]^{m \times n}$  then we have a  $[0, 1]$ -**(bimatrix) game** (aka a **normalized bimatrix game**). The special case of bimatrix games in which all elements of

the bimatrix belong to  $\{0, 1\} \times \{0, 1\}$ , is called a  $\{0, 1\}$ -**(bimatrix) game** (aka a win lose game). A  $\{0, 1\}$ -bimatrix game having (for some integer  $\lambda \geq 1$ ) at most  $\lambda$   $(1, 0)$ -elements per row and at most  $\lambda$  number  $(0, 1)$ -element per column of the bimatrix, is called a  $\lambda$ -**sparse game**. A bimatrix game  $\langle A, B \rangle$  is called **zero sum**, if it happens that  $B = -A$ . In that case the game is solvable in polynomial time, since the two players' optimization problems form a primal-dual linear programming pair [8, Ch.13.2]. In all cases of bimatrix games we assume wlog<sup>2</sup> that  $2 \leq m \leq n$ .

Any probability distribution on the action set  $[m]$  of the row player, ie, any point  $\mathbf{x} \in \Delta_m$ , is a **mixed strategy** for her. Ie, the row player determines her action independently from the column player, according to the probability distribution determined by  $\mathbf{x}$ . Similarly, any point  $\mathbf{y} \in \Delta_n$  is a mixed strategy for the column player. Each extreme point  $\mathbf{e}_i \in \Delta_m$  ( $\mathbf{e}_j \in \Delta_n$ ) that enforces the use of the  $i$ -th row ( $j$ -th column) by the row (column) player, is called a **pure strategy** for her. Any element  $(\mathbf{x}, \mathbf{y}) \in \Delta_m \times \Delta_n$  is a (mixed in general) **strategy profile** for the players. We now define the set of *approximate best responses* for the two players, that will help us simplify the forthcoming definitions:

**Definition 1 (Approximate Best Responses).** Fix any constant  $\varepsilon \geq 0$ , any bimatrix game  $\langle A, B \rangle$  and any profile  $(\mathbf{x}, \mathbf{y})$ . The sets of  $\varepsilon$ -**approximate best responses** and  $\varepsilon$ -**approximate pure best responses** of the row player are defined as:  $BR(\varepsilon, A, \mathbf{y}) \equiv \{\mathbf{x} \in \Delta_m : \mathbf{x}^T A \mathbf{y} \geq \mathbf{z}^T A \mathbf{y} - \varepsilon, \forall \mathbf{z} \in \Delta_m\}$  and  $PBR(\varepsilon, A, \mathbf{y}) \equiv \{i \in [m] : A^i \mathbf{y} \geq A^r \mathbf{y} - \varepsilon, \forall r \in [m]\}$  respectively. The sets of *approximate (pure) best responses* of the column player are defined as:  $BR(\varepsilon, B^T, \mathbf{x}) \equiv \{\mathbf{y} \in \Delta_n : \mathbf{y}^T B^T \mathbf{x} \geq \mathbf{z}^T B^T \mathbf{x} - \varepsilon, \forall \mathbf{z} \in \Delta_n\}$  and  $PBR(\varepsilon, B^T, \mathbf{x}) \equiv \{j \in [n] : B_j^T \mathbf{x} \geq B_r^T \mathbf{x} - \varepsilon, \forall r \in [n]\}$ .

We now provide the definition of Nash Equilibria [19] wrt<sup>3</sup> bimatrix games:

**Definition 2 (Nash Equilibrium).** For any bimatrix game  $\langle A, B \rangle$ , a profile  $(\mathbf{x}, \mathbf{y})$  is a **Nash Equilibrium** (NE in short) iff  $\mathbf{x} \in BR(0, A, \mathbf{y})$  and  $\mathbf{y} \in BR(0, B^T, \mathbf{x})$ . Equivalently,  $supp(\mathbf{x}) \subseteq PBR(0, A, \mathbf{y})$  and  $supp(\mathbf{y}) \subseteq PBR(0, B^T, \mathbf{x})$ .

Due to the apparent difficulty in computing NE for arbitrary bimatrix games, the recent trend is to look for approximate equilibria. The definition of the approximate equilibria is then as follows:

**Definition 3 (Approximate Nash Equilibria).** For any  $\varepsilon \geq 0$  and any bimatrix game  $\langle A, B \rangle$ , a profile  $(\mathbf{x}, \mathbf{y})$  is: (a) An  $\varepsilon$ -**approximate Nash Equilibrium** ( $\varepsilon$ -*ApproxNE* in short) iff each player chooses an  $\varepsilon$ -approximate best response:  $[\mathbf{x} \in BR(\varepsilon, A, \mathbf{y})] \wedge [\mathbf{y} \in BR(\varepsilon, B^T, \mathbf{x})]$ . (b) An  $\varepsilon$ -**well-supported Nash Equilibrium** ( $\varepsilon$ -*SuppNE* in short) iff each player assigns positive probability only to  $\varepsilon$ -approximate pure best responses:  $supp(\mathbf{x}) \subseteq PBR(\varepsilon, A, \mathbf{y})$  and  $supp(\mathbf{y}) \subseteq PBR(\varepsilon, B^T, \mathbf{x})$ .

<sup>2</sup> Without loss of generality.

<sup>3</sup> With respect to.

Clearly, any NE is both a 0-ApproxNE and a 0-SuppNE. On the other hand, every  $\varepsilon$ -SuppNE is also  $\varepsilon$ -ApproxNE, but not necessarily vice versa. Indeed, the only thing we currently know is that from arbitrary  $\frac{\varepsilon^2}{8n}$ -ApproxNE one can construct an  $\varepsilon$ -SuppNE in polynomial time [5]. It is also known that both  $\varepsilon$ -ApproxNE and  $\varepsilon$ -SuppNE are not affected by shifting of the game by some real constant. In Lemma 1 we refine this by showing that the addition of arbitrary row vector  $\mathbf{r}^T$  to all the rows of  $A$  and the addition of arbitrary column vector  $\mathbf{c}$  to all the columns of  $B$ , does not affect the SuppNE of the game.

**Remark:** Note that both notions of approximate equilibria are defined wrt an *additive* error term  $\varepsilon$ . Although (exact) NE are known not to be affected by any positive scaling, it is important to mention that approximate notions of NE are indeed affected. Therefore, from now on we adopt the commonly used assumption in the literature (eg, [18,10,15,4,5,16]) that, when referring to  $\varepsilon$ -ApproxNE or  $\varepsilon$ -SuppNE, the bimatrix game is considered to be a  $[0, 1]$ -bimatrix game.

Of particular importance are the so-called uniform strategies (and profiles) for the players, initially proposed by [18]. The definition is as follows:

**Definition 4 (Uniform Profiles).** A point  $\mathbf{x} \in \Delta_r(k)$  is called a  $k$ -uniform strategy. If  $\mathbf{x} \in \hat{\Delta}_r(k)$  then we refer to a strict  $k$ -uniform strategy. A profile  $(\mathbf{x}, \mathbf{y}) \in \Delta_m \times \Delta_n$  for which  $\mathbf{x}$  is a (strict)  $k$ -uniform strategy and  $\mathbf{y}$  is a (strict)  $\ell$ -uniform strategy, is called a (strict)  $(k, \ell)$ -uniform profile.

We shall denote by  $k$ -NASH the problem of constructing an arbitrary NE for a finite  $k$ -player game in normal form.

### 3 Related Work

The computability of NE in bimatrix games has been a long standing open problem for many years. The most popular algorithm for computing NE in these games, is the algorithm of Lemke and Howson [17]. Unfortunately, it has been recently proved that this pivoting algorithm can make an exponential number of steps before finding a NE, no matter which starting point is chosen [22]. It is also well known that various (quite natural) restrictions of the problem (eg, uniqueness, bounds on support sizes, etc) lead to NP-hard problems [13,7].

In view of all the hardness results for 2-NASH mentioned in Section 1, understanding the limitations of the (in)approximability of the problem is quite important. To our knowledge, the first result that provides  $\varepsilon$ -ApproxNE within *subexponential* time, is the work of Lipton et al. [18]. In particular, for any constant  $\varepsilon > 0$ , the authors prove the existence of an  $\varepsilon$ -ApproxNE for arbitrary  $n \times n$  bimatrix games, which additionally is a uniform profile that has supports of size only  $\mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ . This leads to a rather simple subexponential algorithm for constructing  $\varepsilon$ -ApproxNE for  $[0, 1]$ -bimatrix games, by a simple support



enumeration approach, up to sizes  $\left\lceil \frac{\log n}{\varepsilon^2} \right\rceil$  for each player. A completely analogous result has also been proved in [16] also for  $\varepsilon$ -SuppNE, as a simple application of Althöfer’s Approximation Lemma [2]. The support enumeration approach still remains the fastest strategy to date, for the general problem of providing either  $\varepsilon$ -ApproxNE or  $\varepsilon$ -SuppNE for any *constant*  $\varepsilon > 0$ .

With respect to the tractability of a FPTAS for approximate equilibria, [5] proved that providing a FPTAS for 2-NASH is also **PPAD**-complete. Namely, they proved that unless **PPAD**  $\subseteq$  **P**, there is no algorithm that constructs  $\varepsilon$ -ApproxNE in time  $\text{poly}(n, 1/\varepsilon)$ , for any  $\varepsilon = n^{-\Theta(1)}$ . Moreover, they proved that unless **PPAD**  $\subseteq$  **RP**, there is no algorithm that constructs a NE in time  $\text{poly}(n, 1/\sigma)$ , where  $\sigma$  is the size of the deviation of the elements of the bimatrix. This latter result essentially states that even the smooth complexity of the algorithm of Lemke and Howson is not polynomial.

Two parallel results [10,15] made progress in the direction of providing the first  $\varepsilon$ -ApproxNE for  $[0, 1]$ -bimatrix games and some *constant*  $\varepsilon \in [0, 1)$ . In particular, [10] gave a nice and simple 0.5-ApproxNE for  $[0, 1]$ -bimatrix games, involving only two strategies per player. They also proposed an algorithm, which, under a quite interesting graph theoretic conjecture, constructs in polynomial time a non-trivial SuppNE. Unfortunately, the exact status of the conjecture is still unknown. Independently, [15] provided a  $\frac{3}{4}$ -ApproxNE, as well as a parameterized  $\frac{2+\lambda}{4}$ -ApproxNE for arbitrary  $[0, 1]$ -bimatrix games, where  $\lambda$  is the minimum payoff of a player at a NE of the game. The currently best known result is the polynomial-time construction of a 0.38-ApproxNE [11].

To our knowledge, the only polynomial time algorithm for constructing non-trivial SuppNE for arbitrary  $[0, 1]$ -bimatrix games, was very recently presented in [16]. The quality of this SuppNE depends on the girth of the Nash Dynamics graph, in a properly constructed 0, 1-game. Indeed, this technique works particularly well in cases of bimatrix games corresponding to sparse 0, 1-games with large girth, or 0, 1-games of constant girth.

As for random  $[0, 1]$ -bimatrix games, the work of Bárány, Vempala and Vetta [3] consider the case where all the cells of the payoff matrices are (either uniform, or normal) iid<sup>4</sup> random variables in  $[0, 1]$ . They analyze a simple Las Vegas algorithm for finding a NE in such a game, by brute force on the support sizes, starting from smaller ones. The running time of their algorithm is  $\mathcal{O}(m^2 n \log \log n + n^2 m \log \log m)$ , **whp**<sup>5</sup>. In [16] it was shown that a random  $[0, 1]$ -bimatrix game has the (trivial to construct) uniform full mix as an  $\mathcal{O}\left(\sqrt{\frac{\log n}{m}}\right)$ -SuppNE, under very mild assumptions on randomness, which only require that each row (column) of the bimatrix has its cumulative expectation sharply concentrated. Additionally, in the same work it was shown that **whp** a random 0, 1-game either has a pure NE, or a 0.5-SuppNE with support size 2 for both players, (both detectable in polynomial time).

<sup>4</sup> Independent, identically distributed.

<sup>5</sup> With high probability, ie, with probability  $1 - m^{-c}$ , for some constant  $c > 0$ .

## 4 Our Contribution and Roadmap

The contribution of this paper is the following: We initially construct a 0.5–SuppNE for arbitrary  $\{0, 1\}$ –games, exploiting the tractability of zero sum games (cf. Section 5). To our knowledge, this is the first constant SuppNE for arbitrary win lose games. Essentially, we split *evenly* the divergence from a zero sum game, between the two players. Then we solve this zero sum game in polynomial time, using its direct connection to Linear Programming [8, Ch.13.2]. The computed NE of the zero sum game we consider, is indeed proved to be also a 0.5–SuppNE for the initial  $\{0, 1\}$ –game.

Consequently (cf. Section 6) we propose a polynomial time algorithm for constructing a 0.658–SuppNE for any  $[0, 1]$ –bimatrix game. Again we make only one call to a Linear Programming solver. Interestingly, we need to develop a methodology that constructs  $\phi$ –SuppNE for  $\{0, 1\}$ –games, where  $\phi = \frac{\sqrt{5}-1}{2}$  is the golden ratio. This is clearly worse than the one obtained in the previous section, but nevertheless is easily extended to  $[0, 1]$ –bimatrix games, with only a small deterioration. Additionally, it demonstrates that the even split of the divergence from a zero sum game (which is the optimum choice for win lose games) is not the best choice in case of  $[0, 1]$ –bimatrix games in general.

## 5 Construction of a 0.5–SuppNE for $\{0, 1\}$ –Games

In this section we provide a constant SuppNE for any  $\{0, 1\}$ –game, which directly translates to a constant SuppNE for arbitrary  $[0, 1]$ –bimatrix games, if one exploits a nice observation of [10]. First of all we show that additive transformations have no effect on well supported equilibria. An analogous result was shown for exact NE in [14]:

**Lemma 1.** *Fix arbitrary  $[0, 1]$ –bimatrix game  $\langle A, B \rangle$  and any real matrices  $R, C \in \mathbb{R}^{m \times n}$ , such that  $\forall i \in [m], R^i = \mathbf{r}^T \in \mathbb{R}^n$  and  $\forall j \in [n], C_j = \mathbf{c} \in \mathbb{R}^m$ . Then,  $\forall \varepsilon \in (0, 1), \forall (\mathbf{x}, \mathbf{y}) \in \Delta_m \times \Delta_n$ , if  $(\mathbf{x}, \mathbf{y})$  is an  $\varepsilon$ –SuppNE for  $\langle A, B \rangle$  then it is also an  $\varepsilon$ –SuppNE for  $\langle A + R, B + C \rangle$ .*

*Proof.* Suppose that  $(\mathbf{x}, \mathbf{y})$  is an  $\varepsilon$ –SuppNE for  $\langle A, B \rangle$ . We shall prove that it is also a  $\varepsilon$ –SuppNE for  $\langle A + R, B + C \rangle$ . By the definition of SuppNE, it holds that:  $(\mathbf{x}, \mathbf{y}) \in \varepsilon$ –SuppNE(A,B)  $\Leftrightarrow$

$$\begin{aligned} &\Leftrightarrow \begin{cases} \forall i, r \in [m], x_i > 0 \Rightarrow A^i \mathbf{y} \geq A^r \mathbf{y} - \varepsilon \Rightarrow A^i \mathbf{y} + \mathbf{r}^T \mathbf{y} \geq A^r \mathbf{y} + \mathbf{r}^T \mathbf{y} - \varepsilon \\ \forall j, s \in [n], y_j > 0 \Rightarrow B_j^T \mathbf{x} \geq B_s^T \mathbf{x} - \varepsilon \Rightarrow B_j^T \mathbf{x} + \mathbf{c}^T \mathbf{x} \geq B_s^T \mathbf{x} + \mathbf{c}^T \mathbf{x} - \varepsilon \end{cases} \\ &\Leftrightarrow \begin{cases} \forall i, r \in [m], x_i > 0 \Rightarrow (A + R)^i \mathbf{y} \geq (A + R)^r \mathbf{y} - \varepsilon \\ \forall j, s \in [n], y_j > 0 \Rightarrow (B + C)_j^T \mathbf{x} \geq (B + C)_s^T \mathbf{x} - \varepsilon \end{cases} \\ &\Leftrightarrow (\mathbf{x}, \mathbf{y}) \in \varepsilon$$
–SuppNE(A+R,B+C)

Our next theorem constructs the “right” zero sum game that would stand between the two extreme zero sum games  $\langle A, -A \rangle$  and  $\langle -B, B \rangle$ , wrt an arbitrary  $\{0, 1\}$ –game  $\langle A, B \rangle$ .

**Theorem 1.** For arbitrary  $\{0, 1\}$ -bimatrix game  $\langle A, B \rangle$ , there is a polynomial time constructible profile that is a  $\frac{1}{2}$ -SuppNE of the game.

*Proof.* Consider arbitrary  $\{0, 1\}$ -game  $\langle A, B \rangle \in \{(0, 0), (0, 1), (1, 0)\}^{m \times n}$ . We have excluded the  $(1, 1)$ -elements because these are trivial PNE of the game. We transform the bimatrix  $(A, B)$  into the bimatrix  $(R, C)$  by subtracting 0.5 from all the possible payoffs in the bimatrix:  $R = A - \frac{1}{2}E$  and  $C = B - \frac{1}{2}E$ , where  $E = \mathbf{1} \cdot \mathbf{1}^T$ . We already know that this transformation does not affect the quality of a SuppNE (cf. Lemma [II](#)). Therefore,  $\forall \varepsilon \in [0, 1)$ , each  $\varepsilon$ -SuppNE for  $\langle R, C \rangle$  will also be an  $\varepsilon$ -SuppNE for  $\langle A, B \rangle$ .

We observe that the row player would never accept a payoff less than the one achieved by the (exact) Nash equilibrium  $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$  of the (zero sum) game  $\langle R, -R \rangle$ . This is because strategy  $\hat{\mathbf{x}}$  is a maximin strategy for the row player, and thus the row player can achieve a payoff of at least  $\hat{V}_I \equiv \hat{\mathbf{x}}^T R \hat{\mathbf{y}}$  by adopting  $\hat{\mathbf{x}}$ , for any possible column that the column player chooses  $\mathbf{wpp}$ . Similarly, the column player would never accept a profile  $(\mathbf{x}, \mathbf{y})$  with payoff for her less than  $\tilde{V}_{II} \equiv \tilde{\mathbf{x}}^T C \tilde{\mathbf{y}}$ , where  $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$  is the (exact) NE of the zero sum game  $\langle -C, C \rangle$ . So, we already know that any 0-SuppNE for  $\langle R, C \rangle$  should assure payoffs at least  $\hat{V}_I$  and at least  $\tilde{V}_{II}$  for the row and the column player respectively. Clearly,  $(\hat{\mathbf{x}}, \tilde{\mathbf{y}})$  is a  $\max \left\{ \frac{1}{2} - \hat{V}_I, \frac{1}{2} - \tilde{V}_{II} \right\}$ -ApproxNE of the game, but we cannot assure that it is a nontrivial SuppNE of  $\langle R, C \rangle$ . Nevertheless, inspired by this observation, we attempt to set up the right zero sum game that is somehow connected to  $\langle R, C \rangle$ , whose (exact) NE would provide a good SuppNE for  $\langle R, C \rangle$ . Therefore, we consider an arbitrary zero sum game  $\langle D, -D \rangle$ , for which it holds that  $D = R + X \Leftrightarrow X = D - R$  and  $-D = C + Y \Leftrightarrow Y = -(D + C)$  for some  $m \times n$  bimatrix  $(X, Y)$ . Let again  $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in NE(D, -D)$ . Then we have:

$$\begin{aligned} (\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in NE(D, -D) &= NE(R + X, C + Y) \Leftrightarrow \\ &\Leftrightarrow \begin{cases} \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow (R + X)^i \bar{\mathbf{y}} \geq (R + X)^r \bar{\mathbf{y}} \\ \forall j, s \in [n], \bar{y}_j > 0 \Rightarrow (C + Y)_j^T \bar{\mathbf{x}} \geq (C + Y)_s^T \bar{\mathbf{x}} \end{cases} \\ &\Leftrightarrow \begin{cases} \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow R^i \bar{\mathbf{y}} \geq R^r \bar{\mathbf{y}} - [X^i - X^r] \bar{\mathbf{y}} \\ \forall j, s \in [n], \bar{y}_j > 0 \Rightarrow C_j^T \bar{\mathbf{x}} \geq C_s^T \bar{\mathbf{x}} - [Y_j - Y_s]^T \bar{\mathbf{x}} \end{cases} \end{aligned}$$

Let  $Z \equiv -(R + C)$ . Since  $D = R + X = -(-D) = -(C + Y) \Leftrightarrow -Z = R + C = -(X + Y)$ , we can simply set  $X = Y = \frac{1}{2}Z$ , and then we conclude that:

$$(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in NE(D, -D) \Leftrightarrow \begin{cases} \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow R^i \bar{\mathbf{y}} \geq R^r \bar{\mathbf{y}} - \frac{1}{2} \cdot [Z^i - Z^r] \bar{\mathbf{y}} \\ \forall j, s \in [n], \bar{y}_j > 0 \Rightarrow C_j^T \bar{\mathbf{x}} \geq C_s^T \bar{\mathbf{x}} - \frac{1}{2} \cdot [Z_j - Z_s]^T \bar{\mathbf{x}} \end{cases}$$

Observe now that, since  $R, C \in \left\{ \left(-\frac{1}{2}, -\frac{1}{2}\right), \left(-\frac{1}{2}, \frac{1}{2}\right), \left(\frac{1}{2}, -\frac{1}{2}\right) \right\}^{m \times n}$ , any row of  $Z = -(R + C)$  is a vector in  $\{0, 1\}^n$ , and any column of  $Z$  is a vector in  $\{0, 1\}^m$ . But it holds that  $\forall \hat{\mathbf{z}}, \tilde{\mathbf{z}} \in \{0, 1\}^k, \forall \mathbf{w} \in \Delta_k, (\hat{\mathbf{z}} - \tilde{\mathbf{z}})^T \mathbf{w} \leq \mathbf{1}^T \mathbf{w} = 1$ . So we can be sure that  $\forall i, r \in [m], \forall \mathbf{y} \in \Delta_n, [Z^i - Z^r] \mathbf{y} \leq \mathbf{1}^T \mathbf{y} = 1$ , and  $\forall j, s \in [n], \forall \mathbf{x} \in \Delta_m, [Z_j - Z_s]^T \mathbf{x} \leq \mathbf{1}^T \mathbf{x} = 1$ . Therefore we conclude that:

$$(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in NE \left( R + \frac{1}{2}Z, C + \frac{1}{2}Z \right) \Rightarrow \begin{cases} \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow R^i \bar{\mathbf{y}} \geq R^r \bar{\mathbf{y}} - \frac{1}{2} \\ \forall j, s \in [n], \bar{y}_j > 0 \Rightarrow C_j^T \bar{\mathbf{x}} \geq C_s^T \bar{\mathbf{x}} - \frac{1}{2} \end{cases}$$

$$\Rightarrow (\bar{x}, \bar{y}) \in 0.5\text{-SuppNE}(R, C).$$

## 6 SuppNE for $[0, 1]$ -Bimatrix Games

Given our result on  $\{0, 1\}$ -games, applying a Lemma of Daskalakis et al. [10, Lemma 4.6] for constructing  $\frac{1+\varepsilon}{2}$ -SuppNE of a  $[0, 1]$ -bimatrix game  $\langle A, B \rangle$  by any  $\varepsilon$ -SuppNE of a properly chosen  $\{0, 1\}$ -game of the same size, we could directly generalize our result on SuppNE for  $\{0, 1\}$ -bimatrix games to SuppNE for any  $[0, 1]$ -bimatrix game:

**Corollary 1.** *For any  $[0, 1]$ -bimatrix game  $\langle A, B \rangle$ , there is a  $\frac{3}{4}$ -SuppNE that can be computed in polynomial time.*

The question is whether we can do better than that. Indeed we can, but we first have to modify the rationale of the proof of Theorem 2. This way we shall get a weaker SuppNE for  $\{0, 1\}$ -games, which we can nevertheless extend to  $[0, 1]$ -bimatrix games with only a small deterioration. The next theorem demonstrates the parameterized method for  $\{0, 1\}$ -games, which assures a  $\phi$ -SuppNE.

**Theorem 2.** *For any  $\{0, 1\}$ -bimatrix game, there is a polynomial-time constructible  $\varepsilon(\delta)$ -SuppNE for any  $\delta \in (0, 1)$ , where  $\varepsilon(\delta) \leq \max\{\delta, \frac{1-\delta}{\delta}\}$ .*

*Proof.* Again we try to find a zero sum game that lies somehow between  $\langle R, -R \rangle$  and  $\langle -C, C \rangle$  and indeed provides a guaranteed SuppNE for  $\langle R, C \rangle$ . Therefore, we fix a constant  $\delta \in (0, 1)$  (to be determined later). Consequently, we consider the matrix  $Z = -(R + C) \in \{0, 1\}^{m \times n}$ , indicating (with 1s) the elements of the bimatrix  $\langle R, C \rangle$  that are  $(-\frac{1}{2}, -\frac{1}{2})$ -elements (ie, the  $(0, 0)$ -elements of initial bimatrix  $\langle A, B \rangle$ ). All the other elements are 0s. We now consider the zero sum bimatrix game  $\langle R + \delta Z, -(R + \delta Z) \rangle$ , which is solvable in polynomial time (by use of linear programming). We denote with  $(\bar{x}, \bar{y})$  the (exact) NE of this game. Exploiting the definition of NE we have:

$$\begin{aligned} (\bar{x}, \bar{y}) \in NE(R + \delta Z, -(R + \delta Z)) &\Leftrightarrow \\ &\Leftrightarrow \begin{cases} \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow (R + \delta Z)^i \bar{y} \geq (R + \delta Z)^r \bar{y} \\ \forall j, s \in [n], \bar{y}_j > 0 \Rightarrow (-R - \delta Z)_j^T \bar{x} \geq (-R - \delta Z)_s^T \bar{x} \end{cases} \\ &\Leftrightarrow \begin{cases} \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow R^i \bar{y} + \delta Z^i \bar{y} \geq R^r \bar{y} + \delta Z^r \bar{y} \\ \forall j, s \in [n], \bar{y}_j > 0 \Rightarrow (1 - \delta)R_j^T \bar{x} + \delta(R + Z)_j^T \bar{x} \leq (1 - \delta)R_s^T \bar{x} + \delta(R + Z)_s^T \bar{x} \end{cases} \\ &\Leftrightarrow \begin{cases} \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow R^i \bar{y} + \delta Z^i \bar{y} \geq R^r \bar{y} + \delta Z^r \bar{y} \\ \forall j, s \in [n], \bar{y}_j > 0 \Rightarrow -(1 - \delta)R_j^T \bar{x} + \delta C_j^T \bar{x} \geq -(1 - \delta)R_s^T \bar{x} + \delta C_s^T \bar{x} \end{cases} \\ &\Leftrightarrow \begin{cases} \forall i, r \in [m], \bar{x}_i > 0 \Rightarrow R^i \bar{y} \geq R^r \bar{y} - \delta[Z^i - Z^r] \bar{y} \geq R^r \bar{y} - \varepsilon(\delta) \\ \forall j, s \in [n], \bar{y}_j > 0 \Rightarrow C_j^T \bar{x} \geq C_s^T \bar{x} - \frac{1-\delta}{\delta} \cdot [R_s^T - R_j^T] \bar{x} \geq C_s^T \bar{x} - \varepsilon(\delta) \end{cases} \end{aligned}$$

where we exploited the fact that  $C = -(R + Z)$ , and we denote

$$\varepsilon(\delta) \equiv \max_{i, r \in [m], j, s \in [n], \mathbf{x} \in \Delta_m, \mathbf{y} \in \Delta_n} \left\{ \delta \cdot [Z^i - Z^r] \mathbf{y}, \frac{1-\delta}{\delta} \cdot [R_s^T - R_j^T] \mathbf{x} \right\} \quad (1)$$

Obviously, for any  $\delta \in (0, 1]$  it holds that  $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$  is an  $\varepsilon(\delta)$ -SuppNE for  $\langle R, C \rangle$ . We already proved that  $\forall i, r \in [m], \forall \mathbf{y} \in \Delta_n, [Z^i - Z^r]\mathbf{y} \leq \mathbf{1}^T \mathbf{y} = 1$ . Similarly, every column of  $R$  is a vector from  $\{-\frac{1}{2}, \frac{1}{2}\}^m$ . But the difference  $\hat{\mathbf{u}} - \tilde{\mathbf{u}}$  of any vectors  $\hat{\mathbf{u}}, \tilde{\mathbf{u}} \in \{-\frac{1}{2}, \frac{1}{2}\}^m$  is a vector from  $\{-1, 0, 1\}^m$ . Therefore,  $\forall \hat{\mathbf{u}}, \tilde{\mathbf{u}} \in \{-\frac{1}{2}, \frac{1}{2}\}^m, \forall \mathbf{x} \in \Delta_m, (\hat{\mathbf{u}} - \tilde{\mathbf{u}})^T \mathbf{x} \leq \mathbf{1}^T \mathbf{x} = 1$ . So,  $\forall \delta \in (0, 1], \varepsilon(\delta) \leq \max\{\delta, \frac{1-\delta}{\delta}\}$ .

**Remark:** If we simply set  $\delta = \frac{1-\delta}{\delta} = \frac{\sqrt{5}-1}{2}$ , we conclude that  $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$  is a 0.618-SuppNE for  $\langle R, C \rangle$ , and therefore for  $\langle A, B \rangle$ . Of course, this golden ratio SuppNE is inferior to the previously constructed 0.5-SuppNE for  $\{0, 1\}$ -bimatrix games. But it extends nicely to  $[0, 1]$ -games, using the bound of equation (II).

Now we extend our technique for  $\{0, 1\}$ -games to a technique for arbitrary  $[0, 1]$ -bimatrix games:

**Theorem 3.** *For any  $[0, 1]$ -bimatrix game, a  $(\frac{\sqrt{11}}{2} - 1)$ -SuppNE is constructible in polynomial time.*

*Proof.* We transform our initial  $[0, 1]$ -bimatrix game  $\langle A, B \rangle$  to the corresponding  $[-\frac{1}{2}, \frac{1}{2}]$ -bimatrix game  $\langle R, C \rangle$  where  $R = A - \frac{1}{2}E$  and  $C = B - \frac{1}{2}E$ , without affecting the quality of the produced SuppNE. Our steps are in complete analogy as in the proof of Theorem 2.

Let's assume that we look for some  $\zeta$ -SuppNE of  $\langle R, C \rangle$ . It is clear that the existence of any element  $(R, C)_{i,j} \in [\frac{1}{2} - \zeta, \frac{1}{2}] \times [\frac{1}{2} - \zeta, \frac{1}{2}]$  would indicate a (pure) profile  $(\mathbf{e}_i, \mathbf{e}_j)$  that is already a  $\zeta$ -SuppNE. Since these are detectable in time  $\mathcal{O}(nm)$ , we suppose wlog that for each element of the bimatrix  $(R, C)$ , it holds that  $(R_{i,j} < \frac{1}{2} - \zeta) \vee (C_{i,j} < \frac{1}{2} - \zeta)$ .

We again try to find a zero sum bimatrix game that lies between  $\langle R, -R \rangle$ , and  $\langle -C, C \rangle$ , by using a constant  $\delta \in (0, 1]$  (to be determined later). Due to the same reasoning as in the proof of Theorem 2, we conclude that any  $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in 0$ -SuppNE( $R + \delta Z, -(R + \delta Z)$ ) is actually an  $\varepsilon(\delta)$ -SuppNE for  $\langle R, C \rangle$ , where  $\varepsilon(\delta)$  is defined in equation (II), since there was no use (up to that point) of the fact that indeed we had a  $\{0, 1\}$ -bimatrix game.

Now, in order to find an upper bound in  $\varepsilon(\delta)$ , we first observe that  $\forall j, s \in [n], R_s, R_j \in [-\frac{1}{2}, \frac{1}{2}]^m$ , and therefore,  $\forall \mathbf{x} \in \Delta_m, (R_s - R_j)^T \mathbf{x} \leq 1$ . Similarly, for  $Z$  we observe that  $\forall (i, j) \in [m] \times [n]$ ,

$$-\frac{1}{2} \leq R_{i,j}, C_{i,j} < \frac{1}{2} - \zeta \Rightarrow -1 + 2\zeta < Z_{i,j} = -(R_{i,j} + C_{i,j}) \leq 1$$

$$-\frac{1}{2} \leq R_{i,j} < \frac{1}{2} - \zeta \leq C_{i,j} \leq \frac{1}{2} \Rightarrow -1 + \zeta < Z_{i,j} = -(R_{i,j} + C_{i,j}) \leq \zeta$$

So, we conclude that  $Z \in (-1 + \zeta, 1]^{m \times n}$  and therefore,  $\forall i, r \in [m], \forall \mathbf{y} \in \Delta_n, (Z^i - Z^r)\mathbf{y} \leq 1 - (-1 + \zeta) = 2 - \zeta$ . So, we have already proved that  $\forall \delta \in (0, 1], \varepsilon(\delta) \leq \max\{(2 - \zeta)\delta, \frac{1-\delta}{\delta}\}$ . We assure that  $(2 - \zeta)\delta \geq \frac{1-\delta}{\delta}$  by simply choosing  $\delta^* = \frac{\sqrt{8-4\zeta}-1}{2(2-\zeta)}$ . Then  $\varepsilon^* = \varepsilon(\delta^*) \leq \frac{\sqrt{8-4\zeta}-1}{2}$ . We want this to be at most

**Table 1.** Synopsis of results for SuppNE in win lose and normalized bimatrix games

|                   | GT [16]  |                    | LP                        | Random [16]   |
|-------------------|--|--------------------|---------------------------|---|
| <b>Win Lose</b>   | $1 - 2/g$  |                    | 0.5                       | <b>whp</b> , either $\exists PNE$<br>or $\exists 0.5$ -SuppNE<br>with support sizes 2       |
|                   | $\lambda$ -sparse with large<br>girth $g$ , ie, $\lambda = o(g)$ | $o(1)$             |                           |   |
| <b>Normalized</b> | $1 - 1/g$  |                    | $\frac{\sqrt{11}}{2} - 1$ | Uniform full mix is <b>whp</b><br>$\mathcal{O}\left(\sqrt{\frac{\log m}{m}}\right)$ -SuppNE |
|                   | $\lambda$ -sparse win lose<br>image with large<br>girth          | $\frac{1+o(1)}{2}$ |                           |   |

equal to  $\zeta$ , our initial goal. Therefore, we choose  $\zeta \geq \frac{\sqrt{8-4\zeta}-1}{2} \Leftrightarrow 4\zeta^2+8\zeta-7 \geq 0$ . From this last inequality, our best choice is  $\zeta^* = \frac{\sqrt{11}}{2} - 1$  and we are done.

**Remark:** It is worth mentioning that if we had applied our technique to the first algorithm for computing  $0.5$ -SuppNE in  $\{0, 1\}$ -games, then this would lead to a  $0.667$ -SuppNE for the  $[0, 1]$ -bimatrix game  $\langle A, B \rangle$  which is strictly worse than our current result. Ie, equidistribution (between the two players) of the divergence from the zero sum game is not the right choice for the general algorithm.

## 7 Conclusions

In this work we have made substantial progress towards the construction of SuppNE in bimatrix games. Namely, we propose efficient algorithms for the construction of  $0.5$ -SuppNE for win lose bimatrix games and  $0.658$ -SuppNE for arbitrary  $[0, 1]$ -bimatrix games. These are complementary results to a recent work of ours [16], where we provided SuppNE for both  $\{0, 1\}$ - and  $[0, 1]$ -bimatrix games, whose quality depends on the *girth* of the Nash Dynamics graph. That (combinatorial) approach proved to be extremely good in sparse  $\{0, 1\}$ -games with large girth in the Nash Dynamics graph. Our new (LP based) approach proposed in the present paper, always produces a constant SuppNE, regardless of the structure of the Nash Dynamics graph of the game. The summary of results for SuppNE is presented in Table 1.

## References

1. Abbott, T., Kane, D., Valiant, P.: On the complexity of two-player win-lose games. In: Proc. of the 46th IEEE Symp. on Found. of Comp. Sci (FOCS '05), pp. 113–122. IEEE Computer Society Press, Los Alamitos (2005)
2. Althöfer, I.: On sparse approximations to randomized strategies and convex combinations. Linear Algebra and Applications 199, 339–355 (1994)
3. Bárány, I., Vempala, S., Vetta, A.: Nash equilibria in random games. In: Proc. of the 46th IEEE Symp. on Found. of Comp. Sci (FOCS '05), pp. 123–131. IEEE Computer Society Press, Los Alamitos (2005)

4. Chen, X., Deng, X.: Settling the complexity of 2-player nash equilibrium. In: Proc. of the 47th IEEE Symp. on Found. of Comp. Sci (FOCS '06), pp. 261–272. IEEE Computer Society Press, Los Alamitos (2006)
5. Chen, X., Deng, X., Teng, S.H.: Computing nash equilibria: Approximation and smoothed complexity. In: Proc. of the 47th IEEE Symp. on Found. of Comp. Sci (FOCS '06), pp. 603–612. IEEE Computer Society Press, Los Alamitos (2006)
6. Chen, X., Deng, X., Teng, S.H.: Sparse games are hard. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) WINE 2006. LNCS, vol. 4286, pp. 262–273. Springer, Heidelberg (2006)
7. Conitzer, V., Sandholm, T.: Complexity results about nash equilibria. In: Proc. of the 18th Int. Joint Conf. on Art. Intel (IJCAI '03), pp. 765–771. Morgan Kaufmann, San Francisco (2003)
8. Dantzig, G.B.: Linear Programming and Extensions. Princeton University Press, Princeton, NJ (1963)
9. Daskalakis, C., Goldberg, P., Papadimitriou, C.: The complexity of computing a nash equilibrium. In: Proc. of the 38th ACM Symp. on Th. of Comp (STOC '06). Assoc. of Comp. Mach, pp. 71–78. ACM Press, New York (2006)
10. Daskalakis, C., Mehta, A., Papadimitriou, C.: A note on approximate equilibria. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) WINE 2006. LNCS, vol. 4286, pp. 297–306. Springer, Heidelberg (2006)
11. Daskalakis, C., Mehta, A., Papadimitriou, C.: Progress in approximate nash equilibrium. In: Proc. of the 8th ACM Conf. on El. Commerce (EC '07) (to appear, 2007)
12. Daskalakis, C., Papadimitriou, C.: Three player games are hard. Technical Report TR05-139, Electr. Coll. on Comp. Compl. (ECCC) (2005)
13. Gilboa, I., Zemel, E.: Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior* 1, 80–93 (1989)
14. Kannan, R., Theobald, T.: Games of fixed rank: A hierarchy of bimatrix games. In: Proc. of the 18th ACM-SIAM Symp. on Discr. Alg (SODA '07) (2007)
15. Kontogiannis, S., Panagopoulou, P., Spirakis, P.: Polynomial algorithms for approximating nash equilibria in bimatrix games. In: Spirakis, P.G., Mavronicolas, M., Kontogiannis, S.C. (eds.) WINE 2006. LNCS, vol. 4286, pp. 286–296. Springer, Heidelberg (2006)
16. Kontogiannis, S., Spirakis, P.: Well supported approximate equilibria in bimatrix games: A graph theoretic approach. Technical report, EU Project DELIS – Dynamically Evolving Large Scale Information Systems (January 2007)
17. Lemke, C.E., Howson Jr., J.T.: Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics* 12, 413–423 (1964)
18. Lipton, R., Markakis, E., Mehta, A.: Playing large games using simple strategies. In: Proc. of the 4th ACM Conf. on El. Commerce (EC '03). Assoc. of Comp. Mach., pp. 36–41. ACM Press, New York (2003)
19. Nash, J.: Noncooperative games. *Annals of Mathematics* 54, 289–295 (1951)
20. Papadimitriou, C.: On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. System Sci.* 48, 498–532 (1994)
21. Christos Papadimitriou. Algorithms, games and the internet. In: Proc. of the 33rd ACM Symp. on Th. of Comp. (STOC '01), pp. 749–753 (2001)
22. Savani, R., von Stengel, B.: Hard-to-solve bimatrix games. *Econometrica* 74(2), 397–429 (2006)

# Equational Systems and Free Constructions

(EXTENDED ABSTRACT)

Marcelo Fiore and Chung-Kil Hur\*

Computer Laboratory, University of Cambridge

**Abstract.** The purpose of this paper is threefold: to present a general abstract, yet practical, notion of equational system; to investigate and develop a theory of free constructions for such equational systems; and to illustrate the use of equational systems as needed in modern applications, specifically to the theory of substitution in the presence of variable binding and to models of name-passing process calculi.

## 1 Introduction

The import of equational theories in theoretical computer science is by now well established. Traditional applications include the initial algebra approach to the semantics of computational languages and the specification of abstract data types pioneered by the ADJ group [11], and the abstract description of powerdomain constructions as free algebras of non-determinism advocated by Plotkin [13,16] (see also [1]). While these developments essentially belong to the realm of universal algebra, more recent applications have had to be based on the more general categorical algebra. Examples include theories of abstract syntax with variable binding [6,8], the algebraic treatment of computational effects [17,18], and models of name-passing process calculi [5,21].

In the above and most other applications of equational theories, the existence and construction of initial and/or free algebras, and consequently of monads, plays a central role; as so does the study of categories of algebras. These topics are investigated here in the context of *equational systems*, a very broad notion of equational theories. Examples of equational systems include enriched algebraic theories [14,20], algebras for a monad, monoids in a monoidal category, *etc.* (see Section 3).

The original motivation for the development of the theory of equational systems arose from the need of a mathematical theory readily applicable to two further examples of equational systems: (i)  $\Sigma$ -monoids (see Section 6.1), which are needed for the initial algebra approach to the semantics of languages with variable binding and capture-avoiding simultaneous substitution [6]; and (ii)  $\pi$ -algebras (see Section 6.2), which provide algebraic models of the finitary  $\pi$ -calculus [21]. Indeed, these two examples respectively highlight two inadequacies of enriched algebraic theories in applications: (i) the explicit presentation of an enriched algebraic theory may be hard to give, as it is the case with

---

\* Research supported by a Samsung Scholarship from the Samsung Foundation of Culture.



$\Sigma$ -monoids; and (ii) models may require a theory based on more than one enrichment, as it is the case with  $\pi$ -algebras.

Further benefits of equational systems over enriched algebraic theories are that the theory can be developed for cocomplete, not necessarily locally presentable, categories (examples of which are the category of topological spaces, the category of directed-complete posets, and the category of complete semi-lattices), and that the concept of equational system is straightforwardly dualizable: a co-equational system on a category is simply an equational system on the opposite category (thus, for instance, comonoids in a monoidal category are coalgebras for a co-equational system). On the other hand, the price paid for all this generality is that the important connection between enriched algebraic theories and enriched Lawvere theories [19] is lost for equational systems.

An equational system  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$  is defined as a parallel pair  $L, R : F\text{-Alg} \rightarrow D\text{-Alg}$  of functors between categories of algebras over a base category  $\mathcal{C}$ . In this context, the endofunctor  $F$  on  $\mathcal{C}$ , which generalizes the notion of algebraic signature, is called a functorial signature; the functors  $L, R$  over  $\mathcal{C}$ , generalize the notion of equation, and are called functorial terms; the endofunctor  $D$  on  $\mathcal{C}$  corresponds to the arity of the equation. The category of  $\mathbb{S}$ -algebras is the equalizer  $\mathbb{S}\text{-Alg} \hookrightarrow F\text{-Alg}$  of  $L, R$ . Thus, an  $\mathbb{S}$ -algebra is an  $F$ -algebra  $(X, s : FX \rightarrow X)$  such that  $L(X, s) = R(X, s)$  as  $D$ -algebras on  $X$ .

We have learnt during the course of this work that variations on the concept of equational system have already been considered in the literature. For instance, Fokkinga [7] introduces the more general concept of law between transformers, but only studies initial algebras for the laws that are equational systems; Cirstea [3] introduces the concept of coequation between abstract cosignatures, which is equivalent to our notion of coequational system, and studies final coalgebras for them; Ghani, Lüth, De Marchi, and Power [10] introduce the concept of functorial co-equational presentations, which is equivalent to our notion of coequational system on a locally presentable base category with an accessible functorial signature and an accessible arity endofunctor, and study cofree constructions for them.

Our theory of equational systems (and its dual), which we present in Sections 4 and 5, is more general and comprehensive than that of [7] and [3]; and we relate it to that of [10] in the Concluding Remarks (Section 7).

Free constructions for equational systems are investigated in Section 4. For an equational system  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$ , the existence of free  $\mathbb{S}$ -algebras on objects in  $\mathcal{C}$  is considered in two stages: (i) the construction of free  $F$ -algebras on objects in  $\mathcal{C}$ , and (ii) the construction of free  $\mathbb{S}$ -algebras over  $F$ -algebras. The former captures the construction of freely generated terms with operations from the functorial signature  $F$ ; the latter that of quotienting  $F$ -algebras by the equation  $L = R$  and congruence rules. We give two sufficient conditions for the existence of free  $\mathbb{S}$ -algebras on  $F$ -algebras. The first condition can be used to deduce the existence of free algebras for enriched algebraic theories, but it applies more generally. The second condition may be applied to functorial signatures and arities that are not accessible. The proofs of these results provide constructions of free algebras that may lead to explicit descriptions. As a concrete

example of this situation, we observe that for the functorial signature  $\Sigma_\lambda$  of the  $\lambda$ -calculus, the initial  $\Sigma_\lambda$ -monoid satisfying  $\beta, \eta$  equations consists of  $\lambda$ -terms (up to  $\alpha$ -equivalence) quotiented by the  $\beta, \eta$  equalities (see Section 6.1).

Monads and categories of algebras for equational systems are discussed in Section 5. In the vein of the above results, we give two sufficient conditions under which the monadicity and cocompleteness of categories of algebras follow. As an application, we observe that the category of  $\pi$ -algebras is monadic and cocomplete (see Section 6.2).

## 2 Algebraic Equational Theories

To set our work in context, we briefly review the classical concept of algebraic equational theory and some aspects of the surrounding theory.

An algebraic equational theory consists of a signature defining its operations and a set of equations describing the axioms that it should obey.

A signature  $\Sigma$  is given by a set of operators  $O$  together with a function  $|-| : O \rightarrow \mathbb{N}$  giving an arity to each operator. The set of terms  $T_\Sigma(V)$  on a set of variables  $V$  is built up from the variables and the operators of the signature  $\Sigma$  by the following grammar

$$t \in T_\Sigma(V) ::= v \mid o(t_1, \dots, t_k)$$

where  $v \in V$ ,  $o$  is an operator of arity  $k$ , and  $t_i \in T_\Sigma(V)$  for all  $i = 1, \dots, k$ .

An equation of arity  $V$ , written  $V \vdash l = r$ , for a signature  $\Sigma$  is a pair of terms  $l, r \in T_\Sigma(V)$ .

An algebraic equational theory  $\mathbb{T} = (\Sigma, E)$  is given by a signature  $\Sigma$  together with a set of equations  $E$ .

An algebra for a signature  $\Sigma$  is a pair  $(X, \llbracket - \rrbracket_X)$  consisting of a carrier set  $X$  together with interpretation functions  $\llbracket o \rrbracket_X : X^{|o|} \rightarrow X$  for each operator  $o$  in  $\Sigma$ . By structural induction, such an algebra induces interpretations  $\llbracket t \rrbracket_X : X^V \rightarrow X$  of terms  $t \in T_\Sigma(V)$  as follows:

$$\llbracket t \rrbracket_X = \begin{cases} X^V \xrightarrow{\pi_v} X & , \text{ for } t = v \in V \\ X^V \xrightarrow{\langle \llbracket t_1 \rrbracket_X, \dots, \llbracket t_k \rrbracket_X \rangle} X^k \xrightarrow{\llbracket o \rrbracket_X} X & , \text{ for } t = o(t_1, \dots, t_k) \end{cases}$$

An algebra for the theory  $\mathbb{T} = (\Sigma, E)$  is an algebra for the signature  $\Sigma$  that satisfies the constraints given by the equations in  $E$ , where a  $\Sigma$ -algebra  $(X, \llbracket - \rrbracket_X)$  is said to satisfy the equation  $V \vdash l = r$  whenever  $\llbracket l \rrbracket_X \mathbf{x} = \llbracket r \rrbracket_X \mathbf{x}$  for all  $\mathbf{x} \in X^V$ .

An homomorphism of  $\mathbb{T}$ -algebras from  $(X, \llbracket - \rrbracket_X)$  to  $(Y, \llbracket - \rrbracket_Y)$  is a function  $h : X \rightarrow Y$  between their carrier sets that commutes with the interpretation of each operator; that is, such that  $h(\llbracket o \rrbracket_X(x_1, \dots, x_k)) = \llbracket o \rrbracket_Y(h(x_1), \dots, h(x_k))$ . Algebras and homomorphisms form the category  $\mathbb{T}\text{-Alg}$ .

The existence of free algebras for algebraic theories is one of the most significant properties that they enjoy. For an algebraic theory  $\mathbb{T} = (\Sigma, E)$ , the free algebra over a set  $X$  has as carrier the set  $T_\Sigma(X) / \sim_E$  of equivalence classes of terms on  $X$

under the equivalence relation  $\sim_E$  defined by setting  $t \sim_E t'$  iff  $t$  is provably equal to  $t'$  by the equations given in  $E$  and the congruence rules. The interpretation of each operator on  $T_\Sigma(X)/\sim_E$  is given syntactically:  $\llbracket o \rrbracket([t_1]_{\sim_E}, \dots, [t_k]_{\sim_E}) = [o(t_1, \dots, t_k)]_{\sim_E}$ . This construction gives rise to a left adjoint  $F_{\mathbb{T}}$  to the forgetful functor  $U_{\mathbb{T}} : \mathbb{T}\text{-Alg} \rightarrow \mathbf{Set}$ . Moreover, the adjunction is monadic:  $\mathbb{T}\text{-Alg}$  is equivalent to the category of algebras for the associated monad on  $\mathbf{Set}$ .

We recall the notion of algebra for an endofunctor and how it generalizes that of algebra for a signature.

An algebra for an endofunctor  $F$  on a category  $\mathcal{C}$  is a pair  $(X, s)$  of a carrier object  $X$  in  $\mathcal{C}$  together with a structure algebra map  $s : FX \rightarrow X$ . A homomorphism of  $F$ -algebras from  $(X, s)$  to  $(Y, t)$  is a map  $h : X \rightarrow Y$  in  $\mathcal{C}$  such that  $h \cdot s = t \cdot Fh$ .  $F$ -algebras and homomorphisms form the category  $F\text{-Alg}$ , and the forgetful functor  $U_F : F\text{-Alg} \rightarrow \mathcal{C}$  maps an  $F$ -algebra  $(X, s)$  to its carrier object  $X$ .

As it is well-known, every signature can be turned into an endofunctor on  $\mathbf{Set}$  preserving its algebras. Indeed, for a signature  $\Sigma$ , one defines the corresponding endofunctor as  $F_\Sigma(X) = \coprod_{o \in \Sigma} X^{|o|}$ , so that  $\Sigma\text{-Alg}$  and  $F_\Sigma\text{-Alg}$  are isomorphic. In this view, we will henceforth take endofunctors as a general abstract notion of signature.

**Definition 2.1 (Functorial signature).** *A functorial signature on a category is an endofunctor on it.*

### 3 Equational Systems

We motivate and subsequently present an abstract notion of equation for functorial signatures, leading to the concept of equational system. Free constructions for equational systems are considered in the following section.

Let  $t \in T_\Sigma(V)$  be a term on a set of variables  $V$  for a signature  $\Sigma$ . Recall from the previous section that for every  $\Sigma$ -algebra  $(X, \llbracket - \rrbracket_X)$ , the term  $t$  gives an interpretation function  $\llbracket t \rrbracket_X : X^V \rightarrow X$ . Thus, the term  $t$  determines a function  $\tilde{t}$  assigning to a  $\Sigma$ -algebra  $(X, \llbracket - \rrbracket_X)$  the  $D$ -algebra  $(X, \llbracket t \rrbracket_X)$ , for  $D$  the endofunctor  $(-)^V$  on  $\mathbf{Set}$ . Note that the function  $\tilde{t}$  does not only preserves carrier objects but, furthermore, by the uniformity of the interpretation of terms, that a  $\Sigma$ -homomorphism  $(X, \llbracket - \rrbracket_X) \rightarrow (Y, \llbracket - \rrbracket_Y)$  is also a  $D$ -homomorphism  $(X, \llbracket t \rrbracket_X) \rightarrow (Y, \llbracket t \rrbracket_Y)$ . In other words, the function  $\tilde{t}$  extends to a functor  $\Sigma\text{-Alg} \rightarrow D\text{-Alg}$  over  $\mathbf{Set}$ , i.e. a functor preserving carrier objects and homomorphisms. These considerations lead us to define abstract notions of term and equations as follows.

**Definition 3.1 (Functorial terms and equations).** *A functorial term  $T$  of arity  $D$  for a functorial signature  $F$  on a category  $\mathcal{C}$ , consists of an endofunctor  $D$  on  $\mathcal{C}$  and a functor  $T : F\text{-Alg} \rightarrow D\text{-Alg}$  over  $\mathcal{C}$ , that is, a functor such that  $U_D \cdot T = U_F$ . A functorial equation is given by a pair of functorial terms of the same arity.*

We are now ready to define equational systems, our abstract notion of equational theory.

**Definition 3.2 (Equational systems).** *An equational system*

$$\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$$

*is given by a category  $\mathcal{C}$  and a functorial equation  $L = R$  of arity  $D$  for a functorial signature  $F$ .*

We have restricted attention to equational systems subject to a single equation. The consideration of multi-equational systems  $(\mathcal{C} \triangleright F \vdash \{L_i = R_i : D_i\}_{i \in I})$  subject to a set of equations in what follows is left to the interested reader. We remark however that our development is typically without loss of generality; as, whenever  $\mathcal{C}$  has  $I$ -indexed coproducts, a multi-equational system as above can be expressed as the equational system  $(\mathcal{C} \triangleright F \vdash [L_i]_{i \in I} = [R_i]_{i \in I} : \coprod_{i \in I} D_i)$  with a single equation.

Recall that an equation  $l = r$  in an algebraic theory is interpreted as the constraint that the interpretation functions associated with the terms  $l$  and  $r$  coincide. Hence, for an equational system  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$ , it is natural to say that an  $F$ -algebra  $A$  satisfies the functorial equation  $L = R$  whenever  $L(A) = R(A)$ , and consequently define the category of algebras for the equational system as the full subcategory of  $F\text{-Alg}$  consisting of the  $F$ -algebras that satisfy the functorial equation  $L = R$ . Equivalently, we introduce the following definition.

**Definition 3.3.** *For an equational system  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$ , the category  $\mathbb{S}\text{-Alg}$  of  $\mathbb{S}$ -algebras is the equalizer of  $L, R : F\text{-Alg} \rightarrow D\text{-Alg}$  (in the large category of locally small categories over  $\mathcal{C}$ ).*

Examples of equational systems together with their induced categories of algebras follow.

1. The equational system  $\mathbb{S}_{\mathbb{T}}$  associated to the algebraic theory  $\mathbb{T} = (\Sigma, E)$  is given by  $(\mathbf{Set} \triangleright F_{\mathbb{T}} \vdash L_{\mathbb{T}} = R_{\mathbb{T}} : D_{\mathbb{T}})$ , with  $F_{\mathbb{T}}X = \coprod_{\sigma \in \Sigma} X^{|\sigma|}$ ,  $D_{\mathbb{T}}X = \coprod_{(V \vdash l=r) \in E} X^V$ , and

$$\begin{aligned} L_{\mathbb{T}}(X, \llbracket - \rrbracket_X) &= (X, \llbracket [l] \rrbracket_X \big|_{(l=r) \in E}) , \\ R_{\mathbb{T}}(X, \llbracket - \rrbracket_X) &= (X, \llbracket [r] \rrbracket_X \big|_{(l=r) \in E}) . \end{aligned}$$

It follows that  $\mathbb{T}\text{-Alg}$  is isomorphic to  $\mathbb{S}_{\mathbb{T}}\text{-Alg}$ .

2. More generally, consider an enriched algebraic theory  $\mathbb{T} = (\mathcal{C}, B, E, \sigma, \tau)$  on a locally finitely presentable category  $\mathcal{C}$  enriched over a suitable category  $\mathcal{V}$ , see [14]. Recall that this is given by functors  $B, E : |\mathcal{C}_{\text{fp}}| \rightarrow \mathcal{C}_0$  and a pair of morphisms  $\sigma, \tau : FE \rightarrow FB$  between the free finitary monads  $FB$  and  $FE$  on  $\mathcal{C}$  respectively arising from  $B$  and  $E$ . The equational system  $\mathbb{S}_{\mathbb{T}}$  associated to such an enriched algebraic theory  $\mathbb{T}$  is given by  $(\mathcal{C}_0 \triangleright (GB)_0 \vdash \bar{\sigma}_0 = \bar{\tau}_0 : (GE)_0)$ , where  $GB$  and  $GE$  are the free finitary endofunctors on  $\mathcal{C}$  respectively arising from  $B$  and  $E$ , and where  $\bar{\sigma}$  and  $\bar{\tau}$  are respectively the functors corresponding to  $\sigma$  and  $\tau$  by the bijection between morphisms  $FE \rightarrow FB$  and functors  $GB\text{-Alg} \cong \mathcal{C}^{FB} \rightarrow \mathcal{C}^{FE} \cong GE\text{-Alg}$  over  $\mathcal{C}$ . It follows that  $(\mathbb{T}\text{-Alg})_0$  is isomorphic to  $\mathbb{S}_{\mathbb{T}}\text{-Alg}$ .
3. The definition of Eilenberg-Moore algebras for a monad  $\mathbf{T} = (T, \eta, \mu)$  on a category  $\mathcal{C}$  with binary coproducts can be directly encoded as the equational

system  $\mathbb{S}_{\mathbf{T}} = (\mathcal{C} \triangleright T \vdash L = R : D)$  with  $D(X) = X + T^2X$  and

$$L(X, s) = (X, [s \cdot \eta_X, s \cdot \mu_X]) ,$$

$$R(X, s) = (X, [id_X, s \cdot Ts]) .$$

It follows that  $\mathbb{S}_{\mathbf{T}}\text{-Alg}$  is isomorphic to the category  $\mathcal{C}^{\mathbf{T}}$  of Eilenberg-Moore algebras for  $\mathbf{T}$ .

4. The definition of monoid in a monoidal category  $(\mathcal{C}, \otimes, I, \alpha, \lambda, \rho)$  with binary coproducts yields the equational system  $\mathbb{S}_{\text{Mon}(\mathcal{C})} = (\mathcal{C} \triangleright F \vdash L = R : D)$  with  $F(X) = (X \otimes X) + I$ ,  $D(X) = ((X \otimes X) \otimes X) + (I \otimes X) + (X \otimes I)$ , and

$$L(X, [m, e]) = (X, [m \cdot (m \otimes id_X), \lambda_X, \rho_X]) ,$$

$$R(X, [m, e]) = (X, [m \cdot (id_X \otimes m) \cdot \alpha_{X, X, X}, m \cdot (e \otimes id_X), m \cdot (id_X \otimes e)]) .$$

It follows that  $\mathbb{S}_{\text{Mon}(\mathcal{C})}\text{-Alg}$  is isomorphic to the category of monoids and monoid homomorphisms in  $\mathcal{C}$ .

### 4 Free Constructions for Equational Systems

We investigate sufficient conditions for the existence of free algebras for equational systems; that is, for the existence of a left adjoint to the forgetful functor  $U_{\mathbb{S}} : \mathbb{S}\text{-Alg} \rightarrow \mathcal{C}$ , for  $\mathbb{S}$  an equational system. Since, by definition, the forgetful functor  $U_{\mathbb{S}}$  decomposes as  $\mathbb{S}\text{-Alg} \hookrightarrow F\text{-Alg} \xrightarrow{U_F} \mathcal{C}$ , we will concentrate on obtaining a left adjoint to the embedding  $J_{\mathbb{S}}$ . Conditions for the existence of a left adjoint to  $U_F$  have already been studied in the literature (see *e.g.* [2]).

**Theorem 4.1.** *Let  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$  be an equational system. If  $\mathcal{C}$  is cocomplete, and  $F$  and  $D$  preserve colimits of  $\alpha$ -chains for some infinite limit ordinal  $\alpha$ , then the embedding  $\mathbb{S}\text{-Alg} \hookrightarrow F\text{-Alg}$  has a left adjoint.*

**Theorem 4.2.** *Let  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$  be an equational system. If  $\mathcal{C}$  is well-copowered and cocomplete, and  $F$  preserves epimorphisms, then the embedding  $\mathbb{S}\text{-Alg} \hookrightarrow F\text{-Alg}$  has a left adjoint.*

**Corollary 4.1.** *Let  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$  be an equational system. If  $\mathcal{C}$  is cocomplete,  $F$  preserves epimorphisms and colimits of  $\omega$ -chains, and  $D$  preserves epimorphisms, then the embedding  $\mathbb{S}\text{-Alg} \hookrightarrow F\text{-Alg}$  has a left adjoint. Furthermore the free algebra functor is constructed in  $\omega$  steps.*

These results are proved by performing an iterative, possibly transfinite, construction that associates a free  $\mathbb{S}$ -algebra to every  $F$ -algebra. The cocompleteness of the base category allows one to perform the construction, whilst the other conditions guarantee that the process will eventually stop. We present the construction in the simplest case, *viz.* that of Corollary 4.1. To this end, for an  $F$ -algebra  $(X, s)$ , let  $L(X, s) = (X, l : DX \rightarrow X)$  and  $R(X, s) = (X, r : DX \rightarrow X)$ , and consider the following diagram

$$\begin{array}{ccccccc}
 FX & \xrightarrow{Fe_0} & FX_1 & \cdots & FX_i & \xrightarrow{Fe_i} & FX_{i+1} & \cdots & FX' \\
 \downarrow s & \searrow s_0 & & \searrow s_1 & & \searrow s_i & & \searrow s_{i+1} & \downarrow \exists! s' \\
 DX \xrightarrow[l]{r} X & \xrightarrow{e_0} & X_1 & \xrightarrow{e_1} & X_2 & \cdots & X_{i+1} & \xrightarrow{e_{i+1}} & X_{i+2} & \cdots & X'
 \end{array} \tag{1}$$

where  $e_0$  is a coequalizer of  $l, r$  and where  $(e_{i+1}, s_{i+1})$  is a pushout of  $(s_i, Fe_i)$  for all  $i \geq 0$ . Further, let  $X'$  be a colimit of the  $\omega$ -chain  $\langle e_i \rangle$ , so that  $FX'$  is a colimit of the  $\omega$ -chain  $\langle Fe_i \rangle$ , and define the algebra map  $s'$  to be the unique mediating morphism between them. It follows that  $(X', s')$  is a free  $\mathbb{S}$ -algebra on the  $F$ -algebra  $(X, s)$ .

The intuition behind the construction is that of first quotienting the carrier object by the equation  $L = R$ , and then by congruence rules as much as needed. If free algebras are constructed in  $\omega$  steps, then, roughly speaking, they arise by quotienting a finite number of times.

Finally, we remark that in the presence of binary coproducts the problem of finding free algebras reduces to that of finding initial algebras.

**Proposition 4.1.** *Let  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$  be an equational system on a category  $\mathcal{C}$  with binary coproducts. An  $\mathbb{S}$ -algebra is free over  $A \in \mathcal{C}$  iff it is an initial  $\mathbb{S}^A$ -algebra for  $\mathbb{S}^A = (\mathcal{C} \triangleright (A + F) \vdash L \cdot U^A = R \cdot U^A : D)$  where  $U^A$  denotes the forgetful functor  $(A + F)\text{-Alg} \rightarrow F\text{-Alg}$ .*

## 5 Categories of Algebras for Equational Systems

We consider monads and categories of algebras for equational systems, and give some basic applications of our results.

**Theorem 5.1.** *Let  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$  be an equational system with  $\mathcal{C}$  cocomplete.*

1. *If  $F$  and  $D$  preserve colimits of  $\alpha$ -chains for some infinite limit ordinal  $\alpha$ , then the forgetful functor  $U_{\mathbb{S}} : \mathbb{S}\text{-Alg} \rightarrow \mathcal{C}$  is monadic and  $\mathbb{S}\text{-Alg}$  is cocomplete.*
2. *If  $\mathcal{C}$  is well-copowered,  $F$  preserves epimorphisms, and the forgetful functor  $U_{\mathbb{S}} : \mathbb{S}\text{-Alg} \rightarrow \mathcal{C}$  has a left adjoint, then  $U_{\mathbb{S}}$  is monadic and  $\mathbb{S}\text{-Alg}$  is cocomplete.*

**Proposition 5.1.** *Let  $\mathbb{S} = (\mathcal{C} \triangleright F \vdash L = R : D)$  be an equational system. If the functors  $F$  and  $D$  preserve  $\mathbb{I}$ -indexed colimits for a small category  $\mathbb{I}$  and  $U_{\mathbb{S}}$  has a left adjoint, then the induced monad on  $\mathcal{C}$  also preserves  $\mathbb{I}$ -indexed colimits.*

We revisit the examples of equational systems given in Section 3 in the light of the above results.

1. For the equational system  $\mathbb{S}_{\mathbb{T}} = (\mathbf{Set} \triangleright F_{\mathbb{T}} \vdash L_{\mathbb{T}} = R_{\mathbb{T}} : D_{\mathbb{T}})$  representing an algebraic theory  $\mathbb{T}$ , the category  $\mathbb{S}_{\mathbb{T}}\text{-Alg}$  is monadic over  $\mathbf{Set}$  and cocomplete by Theorem 5.1(1); as  $\mathbf{Set}$  is cocomplete and  $F_{\mathbb{T}}$  and  $D_{\mathbb{T}}$  preserve colimits of  $\omega$ -chains.

2. For the equational system  $\mathbb{S}_{\mathbb{T}} = (\mathcal{C}_0 \triangleright (GB)_0 \vdash \bar{\sigma}_0 = \bar{\tau}_0 : (GE)_0)$  representing an enriched algebraic theory  $\mathbb{T} = (\mathcal{C}, B, E, \sigma, \tau)$ , the category  $\mathbb{S}_{\mathbb{T}}\text{-Alg}$  is monadic over  $\mathcal{C}_0$  and cocomplete by Theorem 5.1(1); as  $\mathcal{C}_0$  is locally finitely presentable and thus cocomplete, and  $(GB)_0$  and  $(GE)_0$  are finitary and thus preserve colimits of  $\omega$ -chains. Furthermore, the monad arising from the monadicity of  $\mathbb{S}_{\mathbb{T}}\text{-Alg}$  is finitary by Proposition 5.1 as so are the functors  $(GB)_0$  and  $(GE)_0$ .
3. One may apply Theorem 5.1(1) to the equational system  $\mathbb{S}_{\mathbb{T}}$  representing a monad  $\mathbf{T} = (T, \eta, \mu)$  on a category  $\mathcal{C}$  with binary coproducts as follows. If  $\mathcal{C}$  is cocomplete and  $T$  preserves colimits of  $\omega$ -chains, then  $\mathbb{S}_{\mathbb{T}}\text{-Alg}$  is monadic over  $\mathcal{C}$  and cocomplete.  
 As another example, consider the powerset monad  $\mathbf{P} = (P, \{-\}, \cup)$  on  $\mathbf{Set}$ . Since  $\mathbf{Set}$  is cocomplete and well-copowered, and the powerset functor  $P$  preserves epimorphisms, by Theorem 4.2, the embedding  $\mathbf{Set}^{\mathbf{P}} \hookrightarrow P\text{-Alg}$  has a left adjoint. We also see that the forgetful functor  $\mathbf{Set}^{\mathbf{P}} \rightarrow \mathbf{Set}$  has a left adjoint from the fact that  $\mathbf{P}$  is a monad. Therefore,  $\mathbf{Set}^{\mathbf{P}}$ , which is isomorphic to the category of complete semi-lattices, is cocomplete (by Theorem 5.1(2)).
4. To the equational system  $\mathbb{S}_{\text{Mon}(\mathcal{C})}$  of monoids in a monoidal category  $\mathcal{C}$  with binary coproducts, we can apply Theorem 5.1(1) as follows. If  $\mathcal{C}$  is cocomplete and the tensor product  $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$  preserves colimits of  $\omega$ -chains, then  $\mathbb{S}_{\text{Mon}(\mathcal{C})}\text{-Alg}$  is monadic over  $\mathcal{C}$  and cocomplete.

## 6 Two Applications

We consider applications of equational systems to the theory of abstract syntax supporting variable binding and substitution [6], and to algebraic models of the  $\pi$ -calculus [21].

### 6.1 $\Sigma$ -Monoids

Following [6], we introduce the concept of  $\Sigma$ -monoid, for a functorial signature  $\Sigma$  with a pointed strength, and consider it from the point of view of equational systems. The theory of equational systems is then used to provide an explicit description of free  $\Sigma$ -monoids. We then show that, for  $\Sigma_\lambda$  the functorial signature of the lambda calculus, the  $\beta, \eta$  identities are straightforwardly expressible as functorial equations. The theory of equational systems is further used to relate the arising algebraic models by adjunctions.

Let  $\Sigma$  be a functorial signature on a monoidal category  $\mathcal{C} = (\mathcal{C}, \otimes, I, \alpha, \lambda, \rho)$ . A pointed strength for  $\Sigma$  is a natural transformation

$$\text{st}_{X,(Y,y:I \rightarrow Y)} : \Sigma(X) \otimes Y \rightarrow \Sigma(X \otimes Y)$$

between functors  $\mathcal{C} \times (I/\mathcal{C}) \rightarrow \mathcal{C}$  satisfying coherence conditions similar to those of strength [15]:

$$\begin{aligned} \rho_{\Sigma A} &= \Sigma(\rho_A) \cdot \text{st}_{A,(I,\text{id}_I)} : \Sigma(A) \otimes I \rightarrow \Sigma A \text{ ,} \\ \text{st}_{A,(B \otimes C,(b \otimes c) \cdot \rho_I^{-1})} \cdot \alpha_{\Sigma A,B,C} \\ &= \Sigma(\alpha_{A,B,C}) \cdot \text{st}_{A \otimes B,(C,c)} \cdot (\text{st}_{A,(B,b)} \otimes \text{id}_C) : (\Sigma(A) \otimes B) \otimes C \rightarrow \Sigma(A \otimes (B \otimes C)) \end{aligned}$$

for all  $A \in \mathcal{C}$  and  $(B, b : I \rightarrow B), (C, c : I \rightarrow C) \in I/\mathcal{C}$ .

For a functorial signature  $\Sigma$  with a pointed strength  $\mathbf{st}$  on a monoidal category  $\mathcal{C}$ , the category of  $\Sigma$ -monoids  $\Sigma\text{-Mon}(\mathcal{C})$  has objects given by quadruples  $(X, s, m, e)$  where  $(X, s)$  is a  $\Sigma$ -algebra and  $(X, m, e)$  is a monoid in  $\mathcal{C}$  satisfying the following compatibility law

$$m \cdot (s \otimes \text{id}_X) = s \cdot \Sigma(m) \cdot \mathbf{st}_{X,(X,e)} : \Sigma(X) \otimes X \rightarrow X ;$$

morphisms are maps of  $\mathcal{C}$  which are both  $\Sigma$ -algebra and monoid homomorphisms.

For  $\mathcal{C}$  with binary coproducts, the equational system  $\mathbb{M}_\Sigma$  of  $\Sigma$ -monoids is defined as  $(\mathcal{C} \triangleright F_\Sigma \vdash L_\Sigma = R_\Sigma : D_\Sigma)$ , with  $F_\Sigma X = \Sigma(X) + (X \otimes X) + I$ ,  $D_\Sigma X = ((X \otimes X) \otimes X) + (I \otimes X) + (X \otimes I) + (\Sigma(X) \otimes X)$ , and

$$\begin{aligned} L_\Sigma(X, [s, m, e]) &= (X, [ \quad m \cdot (m \otimes \text{id}_X) \quad , \quad \lambda_X \quad , \quad \rho_X \quad , \quad m \cdot (s \otimes \text{id}_X) \quad ] ) \\ R_\Sigma(X, [s, m, e]) &= (X, [ m \cdot (\text{id}_X \otimes m) \cdot \alpha_{X,X,X} , m \cdot (e \otimes \text{id}_X) , m \cdot (\text{id}_X \otimes e) , s \cdot \Sigma(m) \cdot \mathbf{st}_{X,(X,e)} ] ). \end{aligned}$$

The functoriality of  $L_\Sigma$  and  $R_\Sigma$  follows from the naturality of  $\alpha, \lambda, \rho$ , and  $\mathbf{st}$ . The isomorphism of  $\mathbb{M}_\Sigma\text{-Alg}$  and  $\Sigma\text{-Mon}(\mathcal{C})$  follows trivially from their definitions.

Consequently, one can apply the theory of equational systems developed in this paper to the algebra of  $\Sigma$ -monoids. For instance, by Theorem 4.1 if  $\mathcal{C}$  is cocomplete, and the functorial signature  $\Sigma$  and the tensor product  $\otimes$  preserve colimits of  $\omega$ -chains, then there exists a free  $\Sigma$ -monoid over every object in  $\mathcal{C}$ . While this only shows the existence of free  $\Sigma$ -monoids, when the monoidal structure is closed, we can go further and give an explicit description of free  $\Sigma$ -monoids using the fact that in this case the initial  $\Sigma$ -monoid exists if so does the initial  $(I + \Sigma)$ -algebra  $\mu X. I + \Sigma X$ , and has carrier object  $\mu X. I + \Sigma X$  equipped with an appropriate  $\Sigma$ -monoid structure, see [6]. Indeed, by Proposition 4.1 a free  $\Sigma$ -monoid over  $A \in \mathcal{C}$  is an initial  $\mathbb{M}_\Sigma^A$ -algebra for the equational system  $\mathbb{M}_\Sigma^A = (\mathcal{C} \triangleright (A + F_\Sigma) \vdash L_\Sigma \cdot U_A = R_\Sigma \cdot U_A : D_\Sigma)$ , where  $U_A$  denotes the forgetful functor  $(A + F_\Sigma)\text{-Alg} \rightarrow F_\Sigma\text{-Alg}$ . Furthermore, one can readily establish the isomorphism  $\mathbb{M}_\Sigma^A\text{-Alg} \cong \mathbb{M}_{(A \otimes -) + \Sigma}\text{-Alg}$ , where the pointed strength  $\mathbf{st}'_{X,(Y,y)}$  for  $(A \otimes -) + \Sigma(-)$  is given by the composite

$$\begin{aligned} &((A \otimes X) + \Sigma(X)) \otimes Y \\ &\cong ((A \otimes X) \otimes Y) + \Sigma(X) \otimes Y \xrightarrow{\alpha_{A,X,Y} + \mathbf{st}'_{X,(Y,y)}} (A \otimes (X \otimes Y)) + \Sigma(X \otimes Y) . \end{aligned}$$

Thus, we have the following result.

**Proposition 6.1.** *For  $\mathcal{C}$  a monoidal closed category with binary coproducts, the free  $\Sigma$ -monoid on  $A \in \mathcal{C}$  exists if so does the initial  $(I + (A \otimes -) + \Sigma(-))$ -algebra  $\mu X. I + A \otimes X + \Sigma X$ , and has carrier object  $\mu X. I + A \otimes X + \Sigma X$  equipped with an appropriate  $\Sigma$ -monoid structure.*

As a concrete example, we now consider the  $\lambda$ -calculus. A  $\lambda$ -model [6] is a  $\Sigma_\lambda$ -monoid for the functorial signature  $\Sigma_\lambda X = X^V + X^2$  with a suitable pointed strength on the presheaf category  $\mathbf{Set}^{\mathbb{F}}$ , where  $\mathbb{F}$  is the (essentially small) category of finite sets and functions, equipped with the substitution monoidal structure  $(\bullet, V)$ . The operations of a  $\Sigma_\lambda$ -monoid  $(X, [\mathbf{abs}, \mathbf{app}, \mathbf{sub}, \mathbf{var}] :$



$X^V + X^2 + (X \bullet X) + V \rightarrow X$ ) provide interpretations of  $\lambda$ -abstraction ( $\mathbf{abs} : X^V \rightarrow X$ ), application ( $\mathbf{app} : X^2 \rightarrow X$ ), capture-avoiding simultaneous substitution ( $\mathbf{sub} : X \bullet X \rightarrow X$ ), and variables ( $\mathbf{var} : V \rightarrow X$ ). The initial  $\lambda$ -model has carrier object  $\mu X. V + X^V + X^2$ , and provides an abstract notion of syntax for the  $\lambda$ -calculus. A syntactic description of free  $\Sigma_\lambda$ -monoids has been considered by Hamana in [12].

The  $\beta, \eta$  identities for a  $\lambda$ -model on  $X$  are expressed by the following equations in the internal language

$$\begin{aligned} (\beta) \quad & f : X^V, x : X \vdash \mathbf{app}(\mathbf{abs}(f), x) = \mathbf{sub}(f(x)) : X \\ (\eta) \quad & x : X \vdash \mathbf{abs}(\lambda v : V. \mathbf{app}(x, \mathbf{var} v)) = x : X \end{aligned}$$

where the map  $-\langle = \rangle : X^V \times X \rightarrow X \bullet X$  embeds  $X^V \times X$  into  $X \bullet X$ . These internal equations provide functorial equations on  $\lambda$ -models, and yield a further equational system  $\mathbb{M}_{\Sigma_\lambda/\beta,\eta}$ . From two applications of Corollary 4.1, we obtain the following adjoint situations:

$$\mathbb{M}_{\Sigma_\lambda/\beta,\eta}\text{-Alg} \overset{\leftarrow \perp}{\underset{\rightarrow}{\rightleftarrows}} \mathbb{M}_{\Sigma_\lambda}\text{-Alg} \overset{\leftarrow \perp}{\underset{\rightarrow}{\rightleftarrows}} (\Sigma_\lambda(-) + (- \bullet -) + V)\text{-Alg} \overset{\leftarrow \perp}{\underset{\rightarrow}{\rightleftarrows}} \mathbf{Set}^{\mathbb{I}}$$

Further, by examining the construction (11) for the free  $\mathbb{M}_{\Sigma_\lambda/\beta,\eta}$ -algebra on the initial  $\mathbb{M}_{\Sigma_\lambda}$ -algebra, one sees that the presheaf of ( $\alpha$ -equivalence classes of)  $\lambda$ -terms is first quotiented by the  $\beta, \eta$  identities, and then by the congruence rules for the operations  $\mathbf{abs}$ ,  $\mathbf{app}$ , and  $\mathbf{sub}$  as much as needed. Thus, the initial  $\mathbb{M}_{\Sigma_\lambda/\beta,\eta}$ -algebra is the presheaf of  $\beta, \eta$ -equivalence classes of  $\lambda$ -terms.

### 6.2 Pi-calculus Algebras

We briefly discuss  $\pi$ -algebras, an algebraic model of the finitary  $\pi$ -calculus introduced by Stark in [21], as algebras for an equational system. The existence of free models is deduced from the theory of equational systems.

We need consider the presheaf category  $\mathbf{Set}^{\mathbb{I}}$ , for  $\mathbb{I}$  the (essentially small) category of finite sets and injections, with the symmetric monoidal closed structure  $(1, \otimes, \multimap)$  induced by the symmetric monoidal structure  $(\emptyset, \uplus)$  on  $\mathbb{I}$  by Day’s construction [4].

A  $\pi$ -algebra is an object  $A \in \mathbf{Set}^{\mathbb{I}}$  together with operations choice :  $A^2 \rightarrow A$ , nil :  $1 \rightarrow A$ , out :  $N \times N \times A \rightarrow A$ , in :  $N \times A^N \rightarrow A$ , tau :  $A \rightarrow A$ , and new :  $(N \multimap A) \rightarrow A$  satisfying the equations of [21, Sections 3.1–3.3 and 3.5]. These algebras, and their homomorphisms, form the category  $\mathcal{PI}(\mathbf{Set}^{\mathbb{I}})$ .

The equational theory for  $\pi$ -algebras is expressed entirely in the internal language of  $\mathbf{Set}^{\mathbb{I}}$  (see also [5]). For example, the equation establishing the inactivity of a process that inputs on a restricted channel is given by

$$p : (A^N)^N \vdash \mathbf{new}(\nu(\lambda x : N. \mathbf{in}(x, p x))) = \mathbf{nil} : A$$

where  $\nu : A^N \rightarrow (N \multimap A)$  is the composite

$$A^N \xrightarrow{up_A \, up_N} (N \multimap A)^{N \multimap N} \xrightarrow{\mathbf{id}^{e_N}} (N \multimap A)^1 \xrightarrow{\cong} (N \multimap A)$$

for  $up_X$  and  $e_X$  respectively the monoidal transposes of

$$X \otimes N \xrightarrow{id_X \otimes !} X \otimes 1 \xrightarrow{\cong} X \quad \text{and} \quad 1 \otimes X \xrightarrow{\cong} X .$$

All these internal equations yield functorial equations, and induce an equational system  $\mathbb{S}_\pi$ .

Since every endofunctor of  $\mathbb{S}_\pi$  is finitary, the following result follows from Theorem 5.1(II).

**Proposition 6.2.** *The category of  $\pi$ -algebras  $\mathcal{PI}(\mathbf{Set}^{\mathbb{I}}) \cong \mathbb{S}_\pi\text{-Alg}$  is cocomplete and monadic over  $\mathbf{Set}^{\mathbb{I}}$ .*

The above discussion also applies more generally, to axiomatic settings as in [5] and, in particular, to  $\pi$ -algebras over the Schanuel topos,  $\omega\mathbf{Cpo}^{\mathbb{I}}$ , etc.

## 7 Concluding Remarks

Our theoretical development also includes the organization of equational systems over a base category into a category. The consideration of colimits, in particular coequalizers, of equational systems led us to introduce the more general concept of *iterated equational system*, for which the whole theory of equational systems generalizes. As an additional result, we have that the category of iterated equational systems over a cocomplete base category is itself cocomplete. This, together with the fact that it embeds the category of accessible monads on the base category as a full subcategory which is closed under colimits, proves that the category of accessible monads on a cocomplete category is also cocomplete. Details will appear elsewhere.

Our theory of equational systems dualizes to one for *coequational systems*. Besides this being of interest in its own right, we note that the proof of the dual of Theorem 4.2, together with the construction of cofree coalgebras for endofunctors by terminal sequences of Worrell [22], gives a construction of cofree coalgebras for coequational systems on a locally presentable base category with an accessible functorial signature that preserves monomorphisms. This result is a variation of a main result of the theory developed by Ghani, Lüth, De Marchi, and Power in [10] (see *e.g.* their Lemmas 5.8 and 5.14); which is there proved by means of the theory of accessible categories without assuming the preservation of monomorphisms but assuming an accessible arity endofunctor.

Ghani and Lüth [9] give an abstract presentation of term rewriting via coinserter in the context of algebraic theories on the category of preorders. In this vein, we have developed a theory of free constructions for *inequational systems* in a preorder-enriched setting, and we are considering applications to higher-order rewriting.

*Acknowledgements.* We are grateful to Sam Staton for discussions.

## References

1. Abramsky, S., Jung, A.: Domain theory. In: Abramsky, S., Gabbay, D.M., Maibaum, T.S.E. (eds.) Handbook of Logic in Computer Science, pp. 1–168. Oxford University Press, Oxford (1994)

2. Barr, M., Wells, C.: *Toposes, Triples and Theories*. Springer, Heidelberg (1985)
3. Cirstea, C.: An algebra-coalgebra framework for system specification. In: Proc. 3rd International Workshop on Coalgebraic Methods in Computer Science. ENTCS, vol. 33, pp. 80–110. Elsevier, Amsterdam (2000)
4. Day, B.: On closed categories of functors. In: Reports of the Midwest Category Seminar IV. LNM, vol. 137, pp. 1–38. Springer, Heidelberg (1970)
5. Fiore, M., Moggi, E., Sangiorgi, D.: A fully abstract model for the  $\pi$ -calculus. *Information and Computation* 179(1), 76–117 (2002)
6. Fiore, M., Plotkin, G., Turi, D.: Abstract syntax and variable binding. In: Proc. 14th IEEE Symp. Logic in Computer Science, pp. 193–202. IEEE Computer Society Press, Los Alamitos (1999)
7. Fokkinga, M.: Datatype laws without signatures. *Mathematical Structures in Computer Science* 6(1), 1–32 (1996)
8. Gabbay, M.J., Pitts, A.: A new approach to abstract syntax with variable binding. *Formal Aspects of Computing* 13, 341–363 (2001)
9. Ghani, N., Lüth, C.: Rewriting via coinserters. *Nordic Journal of Computing* 10(4), 290–312 (2003)
10. Ghani, N., Lüth, C., De Marchi, F., Power, A.J.: Dualising initial algebras. *Mathematical Structures in Computer Science* 13(2), 349–370 (2003)
11. Goguen, J., Thatcher, J., Wagner, E.: An initial algebra approach to the specification, correctness and implementation of abstract data types. In: Yeh, R. (ed.) *Current Trends in Programming Methodology: Software Specification and Design*, vol. IV, chapter 5, pp. 80–149. Prentice Hall, Englewood Cliffs (1978)
12. Hamana, M.: Free  $\Sigma$ -monoids: A higher-order syntax with metavariables. In: Weingan Chin (ed.) *Second Asian Symp. Programming Languages and Systems*. LNCS, vol. 3302, pp. 348–363. Springer, Heidelberg (2004)
13. Hennessy, M., Plotkin, G.: Full abstraction for a simple parallel programming language. In: Becvar, J. (ed.) *Mathematical Foundations of Computer Science*. LNCS, vol. 74, pp. 108–120. Springer, Heidelberg (1979)
14. Kelly, G.M., Power, A.J.: Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads. *Journal of Pure and Applied Algebra* 89, 163–179 (1993)
15. Kock, A.: Strong functors and monoidal monads. *Archiv der Mathematik* 23 (1972)
16. Plotkin, G.: *Domains*. Pisa Notes on Domain Theory (1983)
17. Plotkin, G., Power, A.J.: Algebraic operations and generic effects. *Applied Categorical Structures* 11(1), 69–94 (2003)
18. Plotkin, G., Power, A.J.: Computational effects and operations: An overview. In: Proc. Workshop on Domains VI. ENTCS, vol. 73, pp. 149–163. Elsevier, Amsterdam (2004)
19. Power, A.J.: Enriched Lawvere theories. *Theory and Applications of Categories* 6, 83–93 (1999)
20. Robinson, E.: Variations on algebra: Monadicity and generalisations of equational theories. *Formal Aspects of Computing* 13(3–5), 308–326 (2002)
21. Stark, I.: Free-algebra models for the  $\pi$ -calculus. In: Sassone, V. (ed.) *FOSSACS 2005*. LNCS, vol. 3441, pp. 155–169. Springer, Heidelberg (2005)
22. Worrell, J.: Terminal sequences for accessible endofunctors. In: Proc. 2nd International Workshop on Coalgebraic Methods in Computer Science. ENTCS, vol. 19, Elsevier, Amsterdam (1999)

# Categorical Views on Computations on Trees (Extended Abstract)

Ichiro Hasuo<sup>1</sup>, Bart Jacobs<sup>1</sup>, and Tarmo Uustalu<sup>2</sup>

<sup>1</sup> Institute of Computing and Information Sciences, Radboud University Nijmegen,  
Postbus 9010, NL-6500 GL Nijmegen, The Netherlands

<http://www.cs.ru.nl/~{ichiro,bart}>

<sup>2</sup> Institute of Cybernetics at Tallinn University of Technology,  
Akadeemia tee 21, EE-12618 Tallinn, Estonia

<http://www.cs.ioc.ee/~tarmo>

**Abstract.** Computations on trees form a classical topic in computing. These computations can be described in terms of machines (typically called tree transducers), or in terms of functions. This paper focuses on three flavors of bottom-up computations, of increasing generality. It brings categorical clarity by identifying a category of tree transducers together with two different behavior functors. The first sends a tree transducer to a coKleisli or biKleisli map (describing the contribution of each local node in an input tree to the global transformation) and the second to a tree function (the global tree transformation). The first behavior functor has an adjoint realization functor, like in Goguen’s early work on automata. Further categorical structure, in the form of Hughes’s Arrows, appears in properly parameterized versions of these structures.

## 1 Introduction

Tree transformations are functions sending trees to trees. Such transformations are of broad interest in computing, notably in language processing, and are often studied in relation to certain types of realizing machines. They form a classical topic.

In this paper we aim at a systematic study of phenomena and constructions related to *bottom-up* tree transformations. We first sketch two motivating observations: these will later be given detailed accounts.

*Behavior-realization adjunction.* It is a fundamental idea in computer science that we associate with a “computable” function a “machine” which realizes it. Those machines which realize tree transformations are often called *tree transducers* and have been extensively studied as a continuation of automata theory: see [10,11,2] and also more recently [1].

Here comes our first question. What do we mean by saying “a machine  $c$  realizes a transformation  $l$ ”? Given a transformation  $l$ , is there a machine which realizes it? Is there a canonical choice among such realizers? We shall answer these questions, following the idea of Goguen’s *behavior-realization adjunction* [3] for (a more elementary setting of) automata, see also [9].

*Tree functions from local behaviors.* We start with relabeling bottom-up tree transformations that only change labels on each node of an input tree, like  $l$  on the left.

$$\begin{array}{ccc}
 \begin{array}{c} a_4 \\ / \quad \backslash \\ a_2 \quad a_3 \\ / \quad \backslash \\ a_0 \quad a_1 \end{array} & \xrightarrow{l} & \begin{array}{c} b_4 \\ / \quad \backslash \\ b_2 \quad b_3 \\ / \quad \backslash \\ b_0 \quad b_1 \end{array} & & \begin{array}{c} a_4 \\ / \quad \backslash \\ a_2 \quad a_3 \\ / \quad \backslash \\ a_0 \quad a_1 \end{array} & \xrightarrow{k} & b_4 & (1)
 \end{array}$$

Now let us consider another function  $k$  which operates on the same input trees as  $l$  does but returns the root label of the output tree of  $l$ . That is,  $k = \epsilon \circ l$  where  $\epsilon$  extracts the root label. It may seem that  $k$  (which shall be called a *local behavior*) carries less information than  $l$  does— $\epsilon$  throws information away. But when  $l$  is relabeling bottom-up we can recover  $l$  from  $k$ .

Our main contribution is to give an account of some classes of tree transformations in terms of diagrams like this:

$$\begin{array}{ccc}
 \mathbf{TT} & \xrightarrow{TF} & \mathbf{TF} \\
 \uparrow \text{Real} \quad \downarrow \text{LBeh} & & \uparrow \\
 \mathbf{LBeh} & \xrightarrow{W} & \mathbf{TF}
 \end{array}
 \quad (2)$$

Here,  $TF$  and  $LBeh$  are two *behavior functors* from the category of tree transducers (“machines”) to tree functions and to local behaviors. For relabelings, the functor  $W$  is an isomorphism: this embodies the equivalence of the two behaviors  $TF$  and  $LBeh$  as hinted at above; for more general types of tree transformations, it will be epi. The category  $\mathbf{TF}_\uparrow$  is included in  $\mathbf{TF}$  of tree functions in general: we shall give a categorical characterization of being “bottom-up”. The local behaviors are coKleisli maps of certain comonads, in one case biKleisli maps of a distributive law of a comonad over a monad, and agree with the idea of comonadic notions of computation as those that send “values-in-contexts” to “values” [13,12] (the latter reference deals with attribute grammars, another type of tree computations). The behavior-realization adjunction is presented as  $Real \dashv LBeh$ .

In each of the Sects. 2–4, we shall develop a situation like (2) for a specific class of tree transformations—and hence a corresponding class of tree transducers. Namely, *relabeling bottom-up tree transducers* in Sect. 2; *rebranching bottom-up tree transducers* in Sect. 3; and *bottom-up tree transducers* in full generality in Sect. 4. In Sect. 5 we generalize our categorical formulation in an orthogonal direction: we uncover further compositional structures using Hughes’s Arrows [5], and thus a way to view tree transformations as “structured computations” in programming semantics.

## 2 Relabeling Bottom-Up Tree Transducers

In this section we will consider a class of tree transducers (TTs) that operate on well-founded trees of a fixed branching type  $F$  (a set functor), with labels

at their nodes taken from a parameter set, or alphabet. These transducers take  $A$ -labeled trees to  $B$ -labeled trees for fixed alphabets  $A, B$ , but Section 5 will sketch a properly parameterized version. They work bottom-up by only changing the labels of an input tree and are thus shape-preserving.

For this class of TTs, we shall turn the informal diagram (2) from the introduction into the diagram below. It has: two behavior functors  $LBeh$  and  $TF$ ; a functor  $W$  establishing equivalence of two kinds of behavior; and an adjunction  $Real \dashv LBeh$ .

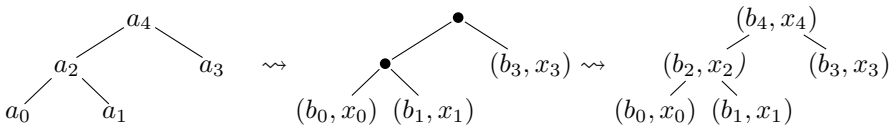
$$\begin{array}{ccc}
 \mathbf{TT}(A, B) & \xrightarrow{TF} & \\
 \text{Real} \left( \begin{array}{c} \uparrow \\ \dashv \\ \downarrow \end{array} \right) LBeh & & \mathbf{TF}_{\uparrow}(A, B) \xrightarrow{\quad} \mathbf{TF}(A, B) \\
 \mathbf{LBeh}(A, B) & \xrightarrow{W} & \cong
 \end{array} \tag{3}$$

That the branching type of our trees is expressed by a set functor  $F$  generalizes more traditional universal-algebraic signatures, given by a set  $\Sigma$  of operations  $f$ , each with an arity  $|f| \in \mathbb{N}$ . Such a signature yields a functor  $Z \mapsto \prod_{f \in \Sigma} Z^{|f|}$ . The  $A$ -labeled trees of the branching type  $F$  (for brevity, we also say  $A$ -trees) live in the initial algebra of the functor  $A \times F_{-}$ , whose carrier we denote by  $DA$ , leaving  $F$  implicit. The algebra structure  $A \times FDA \xrightarrow{\cong} DA$  will be denoted by  $\sigma_A$ . Obviously  $D1$  is the set of unlabelled trees or tree-shapes.

**Definition 2.1.** A (relabeling bottom-up) tree transducer (TT) is a function  $A \times FX \xrightarrow{c} B \times X$  in **Sets**, where the set  $X$  is called the state space. A morphism of such TTs from  $A \times FX \xrightarrow{c} B \times X$  to  $A \times FY \xrightarrow{d} B \times Y$  is a function  $f : X \rightarrow Y$  satisfying  $(B \times f) \circ c = d \circ (A \times Ff)$ .

TTs and morphisms between them form a category which we denote by  $\mathbf{TT}(A, B)$ , leaving again the dependence on  $F$  implicit. Obviously,  $\mathbf{TT}(A, B)$  is nothing but the comma category  $(A \times F_{-} \downarrow B \times _{-})$ .

**Example 2.2.** The operation of a TT is best described on an example. As the branching type  $F$  we take  $1+(\_ )^2$ , describing well-founded binary trees. Consider a TT  $A \times (1 + X^2) \xrightarrow{c} B \times X$  and the leftmost tree below as an input.



The bottom-up computation starts at the leaves: let  $(a_0, \kappa_1(*)) \xrightarrow{c} (b_0, x_0)$ , where  $\kappa_1, \kappa_2$  are coproduct injections. This assigns a label  $b_0$  and a state  $x_0$  to the corresponding leaf of the output tree. Similar mappings at the other leaves lead to the middle tree. At the inner position of  $a_2$ , the label on the output tree is determined by the input label  $a_2$  as well as by the states  $x_0, x_1$  of the successor nodes. They are already available precisely because we proceed in a bottom-up manner. Now we get  $(b_2, x_2)$  from the outcome  $(a_2, \kappa_2(x_0, x_1)) \xrightarrow{c} (b_2, x_2)$ . We continue this way and get the tree on the right from  $(a_4, \kappa_2(x_2, x_3)) \xrightarrow{c} (b_4, x_4)$ .

By forgetting about the states  $x_i$ , we finally obtain the output tree of the computation. It is obvious that the shape of the input tree is preserved. This will change in the next section.

For a TT  $c$ , we shall now define two behaviors  $TF(c)$  and  $LBeh(c)$ . The former is a function that carries an  $A$ -tree to a  $B$ -tree; the latter carries an  $A$ -tree to an element in  $B$ , as hinted at in the introduction.

**Definition 2.3.** A TT  $A \times FX \xrightarrow{c} B \times X$  induces its tree function behavior  $TF(c) : DA \rightarrow DB$  and its local behavior  $LBeh(c) : DA \rightarrow B$  via the following two diagrams, both using the initiality of  $\sigma_A$ .

$$\begin{array}{ccc}
 A \times FDA & \dashrightarrow & A \times F(DB \times X) \\
 \sigma_A \downarrow \cong & & \downarrow \check{c} \\
 DA & \dashrightarrow & DB \times X \\
 & \searrow TF(c) & \downarrow \pi_1 \\
 & & DB
 \end{array}
 \qquad
 \begin{array}{ccc}
 A \times FDA & \dashrightarrow & A \times F(B \times X) \\
 \sigma_A \downarrow \cong & & \downarrow A \times F\pi_2 \\
 DA & \dashrightarrow & A \times FX \\
 & \searrow LBeh(c) & \downarrow c \\
 & & B \times X \\
 & & \downarrow \pi_1 \\
 & & B
 \end{array}$$

where the algebra structure  $\check{c}$  on the left is given by the composite

$$A \times \underline{F(DB \times X)} \xrightarrow{\langle F\pi_1, F\pi_2 \rangle} \underline{A} \times \underline{FDB} \times \underline{FX} \xrightarrow{c} \underline{B} \times \underline{FDB} \times X \xrightarrow{\sigma_B} DB \times X$$

the underlining indicating what the maps act on.

The mapping  $A \mapsto DA$  carries a comonad structure. It is the cofree recursive comonad on  $F$  [14]. A local behavior  $LBeh(c) : DA \rightarrow B$  is a morphism  $A \rightarrow B$  in the coKleisli category of the comonad  $D$ . This is a general phenomenon.

By a simple diagram chase it can be seen that a morphism of TTs is indeed a “behavior-preserving map” wrt. the above two behaviors.

**Lemma 2.4.** Assume we have a morphism  $f$  from one TT  $c$  to another  $d$ . Then  $LBeh(c) = LBeh(d)$  and  $TF(c) = TF(d)$ . □

In Example 2.2 we have illustrated how a TT acts in a bottom-up fashion on trees. Before we can show that the  $TF$  behavior from Def. 2.3 is indeed “bottom-up” we need a characterization of bottom-up tree functions. Intuitively, these are the functions  $l : DA \rightarrow DB$  such that:

$$l \left( \begin{array}{c} a \\ \triangleleft \quad \triangleright \\ t_1 \quad t_2 \end{array} \right) \text{ is of the form } \begin{array}{c} \bullet \\ \triangleleft \quad \triangleright \\ l(t_1) \quad l(t_2) \end{array}$$

The following definition captures this intuition in categorical terms.

**Definition 2.5.** A tree function  $l : DA \rightarrow DB$  is said to be (relabeling) bottom-up if it is a morphism of coalgebras, as in:

$$\begin{array}{ccc}
 FDA & \xrightarrow{Fl} & FDB \\
 \uparrow \pi_2 & & \uparrow \pi_2 \\
 A \times FDA & & B \times FDB \\
 \cong \uparrow \sigma_A^{-1} & \xrightarrow{l} & \cong \uparrow \sigma_B^{-1} \\
 DA & \xrightarrow{\quad} & DB
 \end{array} \tag{4}$$

**Lemma 2.6.** For a TT  $A \times FX \xrightarrow{c} B \times X$ , the induced tree function  $TF(c) : DA \rightarrow DB$  is bottom-up. □

Now we can define the three semantic domains appearing in (3). We write:

- **LBeh**( $A, B$ ) for the set of maps  $DA \rightarrow B$ , i.e.,  $\mathbf{LBeh}(A, B) = \text{Hom}_{\mathbb{C}}(DA, B)$ ;
- **TF**( $A, B$ ) for the set of maps  $DA \rightarrow DB$ , i.e.,  $\mathbf{TF}(A, B) = \text{Hom}_{\mathbb{C}}(DA, DB)$ ;
- $\mathbf{TF}_{\uparrow}(A, B) \hookrightarrow \mathbf{TF}(A, B)$  for the subset of bottom-up maps  $DA \rightarrow DB$ .

These three sets are considered as discrete categories. This enables us to consider behavior mappings as functors from  $\mathbf{TT}(A, B)$ , in a degenerate fashion.

**Lemma 2.7.** The mappings *LBeh* and *TF* in Def. 2.3 extend to functors

$$\mathbf{LBeh} : \mathbf{TT}(A, B) \rightarrow \mathbf{LBeh}(A, B) \quad \text{and} \quad \mathbf{TF} : \mathbf{TT}(A, B) \rightarrow \mathbf{TF}(A, B) .$$

The functor *TF* factors through the embedding  $\mathbf{TF}_{\uparrow}(A, B) \hookrightarrow \mathbf{TF}(A, B)$ . □

A realization functor  $Real : \mathbf{LBeh}(A, B) \rightarrow \mathbf{TT}(A, B)$  is defined to send a local behavior  $k : DA \rightarrow B$  to the TT  $\langle k, DA \rangle \circ \sigma_A : A \times FDA \rightarrow B \times DA$ . This TT has a canonical state space, namely the set  $DA$  of all  $A$ -trees; in all but degenerate cases, this state space is infinite. In fact *Real* yields the initial realization and we get a behavior-realization adjunction in the spirit of [3].

**Theorem 2.8.** We have  $Real \dashv \mathbf{LBeh}$ , and since the category  $\mathbf{LBeh}(A, B)$  is discrete, this adjunction is actually a coreflection.

*Proof.* The statement is equivalent to the following. For a given local behavior  $DA \xrightarrow{k} B$ , the realization  $Real(k)$  is the initial one among those which yield  $k$  as their *LBeh* behavior. Let  $A \times FX \xrightarrow{c} B \times X$  be one of such TTs. The following fact is shown by diagram chasing.

$$\begin{array}{l}
 DA \xrightarrow{f} X \text{ is a morphism of TTs from } Real(k) \text{ to } c \text{ if and only if } \langle k, f \rangle \text{ is} \\
 \text{an algebra homomorphism from the initial algebra } \sigma_A \text{ to } c \circ (A \times F\pi_2) : \\
 A \times F(B \times X) \rightarrow B \times X.
 \end{array}$$

Initiality of  $\sigma_A$  yields existence and uniqueness of such  $f$ , hence the initiality of  $Real(k)$ . □



Next we shall establish an isomorphism between the two (local and tree function) behaviors, which we already discussed in the introduction. By Lemma 2.7, Theorems 2.8 and 2.9 we have established the situation (3).

**Theorem 2.9.** *The following composite  $W$  of functors is an isomorphism.*

$$W = \left( \mathbf{LBeh}(A, B) \xrightarrow{Real} \mathbf{TT}(A, B) \xrightarrow{TF} \mathbf{TF}_\uparrow(A, B) \right)$$

*Proof.* The functor  $W$  sends a map  $k : DA \rightarrow B$  to its coKleisli extension  $Dk \circ \delta_A : DA \rightarrow DDA \rightarrow DB$ . Let  $E : \mathbf{TF}_\uparrow(A, B) \rightarrow \mathbf{LBeh}(A, B)$  be the functor carrying a bottom-up tree function  $l : DA \rightarrow DB$  to  $\epsilon_B \circ l : DA \rightarrow DB \rightarrow B$ . Thus  $E$  post-composes the tree function with extraction of the root label. Then  $E \circ W = \text{Id}$  because  $D$  is a comonad. For the opposite direction  $W \circ E = \text{Id}$ , bottom-upness is crucial.  $\square$

### 3 Rebranching Bottom-Up Tree Transducers

In this section we pursue the same idea as in the previous section, but for a more general class of bottom-up TTs, namely *rebranching TTs*. They no longer preserve tree shapes, in fact they take trees of one branching type  $F$  to trees of a possibly different branching type  $G$ , by reorganizing the branching of any node of the input tree from type  $F$  to type  $G$ .

We shall establish the following situation, which is almost the same as (3). The main differences are: 1) the fixed parameters are now functors  $F, G$  for branching types (instead of sets  $A, B$  of labels) meaning that we consider transformations of  $F$ -branching trees ( $F$ -trees for short) into  $G$ -trees; 2) the isomorphism between  $\mathbf{LBeh}$  and  $\mathbf{TF}_\uparrow$  is not present.

$$\begin{array}{ccc}
 \mathbf{TT}(F, G) & \xrightarrow{TF} & \mathbf{TF}_\uparrow(F, G) \hookrightarrow \mathbf{TF}(F, G) \\
 \begin{array}{c} \mathbf{LBeh}(F, G) \\ \uparrow \text{Real} \left( \dashv \right) \\ \mathbf{LBeh}(F, G) \end{array} & \xrightarrow{W} & \mathbf{TF}_\uparrow(F, G)
 \end{array} \tag{5}$$

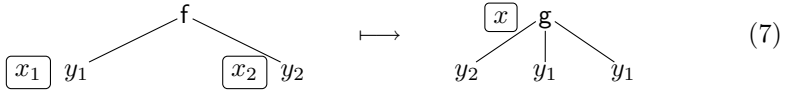
A novelty in this section is what we call “placeholders-via-naturality”. TTs are conventionally systems of transition rules in which placeholders appear explicitly. In our categorical approach, they have quite a different presentation as natural transformations (Def. 3.1). The correspondence between these seemingly different notions will be described via the Yoneda lemma.

Let us first present the conventional notion of rebranching TTs. Let  $\Sigma$  and  $\Delta$  be universal-algebraic signatures: we consider transformations of  $\Sigma$ -trees into  $\Delta$ -trees. Conventionally, a rebranching TT with a state space  $X$  is presented as an element of the set

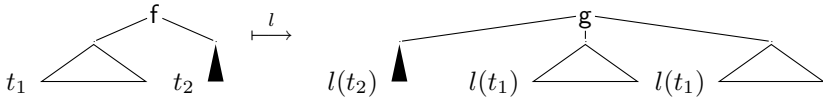
$$\prod_{f \in \Sigma} \left( X^{|f|} \longrightarrow \left( \prod_{g \in \Delta} |f|^{|g|} \right) \times X \right) . \tag{6}$$

It is illustrative to think of the cardinality  $|f|$  as a set  $\{y_1, \dots, y_{|f|}\}$  of placeholders, of the set  $X^{|f|}$  on the left as the set of graphs of functions from  $|f|$  to  $X$  and of the

set  $|f|^{|\mathbf{g}|}$  on the right as the set of length- $|\mathbf{g}|$  lists over  $|f|$ . For example, assume that some  $f$  is binary and a TT (6) carries  $(f, ((y_1, x_1), (y_2, x_2)))$  to  $((\mathbf{g}, (y_2, y_1, y_1)), x)$  with a ternary  $\mathbf{g}$ . This is understood graphically as follows.



This is “bottom-up” because the state  $x$  is determined by the states  $x_1, x_2$  assigned to its successor nodes. Placeholders  $y_1, y_2$  designate how the subtrees are reorganized in the bottom-up construction of a tree function behavior  $l$ .



The name *rebranching* comes from the fact that, on the right hand side of (7), exactly one function symbol occurs, so that a layer in a input tree is sent to exactly one layer of the output tree, and only the branching within the layer changes. In Sect. 4 we will abandon also this requirement.

We now present our categorical definition of TTs.

**Definition 3.1.** A (rebranching bottom-up) TT is a natural transformation  $F(\_ \times X) \xrightarrow{\gamma} G\_ \times X$  between set functors. The set  $X$  is called its state space. A morphism of TTs from  $F(\_ \times X) \xrightarrow{\gamma} G\_ \times X$  to  $F(\_ \times Y) \xrightarrow{\delta} G\_ \times Y$  is a function  $f : X \rightarrow Y$  satisfying  $(G\_ \times f) \circ \gamma = \delta \circ F(\_ \times f)$ . We denote by  $\mathbf{TT}(F, G)$  the category of TTs and morphisms.

This categorical formulation may seem very different from the conventional one (6). But somewhat remarkably the two agree for functors arising from traditional signatures.

Let  $F, G$  be induced by universal-algebraic signatures  $\Sigma, \Delta$ : namely,  $F = \coprod_{f \in \Sigma} (\_)^{|f|}$  and  $G = \coprod_{g \in \Delta} (\_)^{|g|}$ . The following calculation shows the equivalence between (6) and Def. 3.1 via the Yoneda lemma.

$$\begin{aligned}
 & \prod_{f \in \Sigma} (X^{|f|} \rightarrow (\prod_{g \in \Delta} |f|^{|\mathbf{g}|}) \times X) \\
 &= \prod_{f \in \Sigma} (G^{|f|} \times X)^{X^{|f|}} && \text{by definition of } G \\
 &= \prod_{f \in \Sigma} ((\_)^{|f|} \Rightarrow (G\_ \times X)^{X^{|f|}}) && \text{by Yoneda} \\
 &= \prod_{f \in \Sigma} ((\_ \times X)^{|f|} \Rightarrow G\_ \times X) && \text{by } \_ \times X^{|f|} \dashv (\_)^{X^{|f|}} \\
 &= (\prod_{f \in \Sigma} (\_ \times X)^{|f|}) \Rightarrow G\_ \times X \\
 &= F(\_ \times X) \Rightarrow G\_ \times X && \text{by definition of } F .
 \end{aligned}$$

On the third line the set of placeholders (the first occurrence of  $|f|$  on the second line) is absorbed into naturality, hence “placeholders-via-naturality”.

We now proceed to the tree function behavior of our TTs. The tree functions here take  $F$ -trees to  $G$ -trees. Going slightly more general than necessary for this

section (but preparing for the next), we write  $F^*Z$  for the carrier of the initial  $(Z + F_-)$ -algebra, i.e., the set of unlabelled  $F$ -trees with variables (graft-points) from a set  $Z$ . For the algebra structure  $F(F^*Z) \cong F^*Z$  we write  $\alpha_Z^F$ .  $F$ -trees simpliciter (i.e., those without variables) arise as the special case  $F^*0$ . The set (or discrete category) of tree functions  $F^*0 \rightarrow G^*0$  will be denoted by  $\mathbf{TF}(F, G)$ .

**Definition 3.2.** A TT  $F(\_ \times X) \xrightarrow{\tilde{\gamma}} G\_ \times X$  induces its tree-function behavior  $TF(\gamma) \in \mathbf{TF}(F, G)$  by the following algebra initiality diagram.

$$\begin{array}{ccc}
 FF^*0 & \dashrightarrow & F(G^*0 \times X) \\
 \alpha_0^F \downarrow \cong & & \downarrow \gamma_{G^*0} \\
 & & GG^*0 \times X \\
 & & \cong \downarrow \alpha_0^G \times X \\
 F^*0 & \dashrightarrow^{\tilde{\gamma}} & G^*0 \times X \\
 & \searrow TF(\gamma) & \downarrow \pi_1 \\
 & & G^*0
 \end{array} \tag{8}$$

Here again, similarly to the situation for relabelings, not all the tree functions  $F^*0 \rightarrow G^*0$  are induced by a TT but only “bottom-up” ones are.

**Definition 3.3.** A tree function  $F^*0 \xrightarrow{l} G^*0$  is said to be (rebranching) bottom-up, if there exists a natural transformation called a witness  $F(\_ \times F^*0) \xrightarrow{\omega} G\_$  which makes the following diagram commute.

$$\begin{array}{ccc}
 GF^*0 & \xrightarrow{Gl} & GG^*0 \\
 \omega_{F^*0} \uparrow & & \uparrow \\
 F(F^*0 \times F^*0) & & \cong (\alpha_0^G)^{-1} \\
 F\langle \text{id}, \text{id} \rangle \uparrow & & \uparrow \\
 FF^*0 & & \\
 (\alpha_0^F)^{-1} \uparrow \cong & & \\
 F^*0 & \xrightarrow{l} & G^*0
 \end{array} \tag{9}$$

By  $\mathbf{TF}_\uparrow(F, G)$  we denote the set (discrete category) of tree functions  $F^*0 \rightarrow G^*0$  which are rebranching bottom-up. We have  $\mathbf{TF}_\uparrow(F, G) \hookrightarrow \mathbf{TF}(F, G)$ .

Witnesses are not necessarily unique. A simple example is the tree function that sends an unlabelled binary tree to the unlabelled unary tree of its height.

**Lemma 3.4.** For a TT  $F(\_ \times X) \xrightarrow{\tilde{\gamma}} G\_ \times X$ , the induced tree function  $TF(\gamma) : F^*0 \rightarrow G^*0$  is (rebranching) bottom-up.

*Proof.* Take  $\omega = \pi_1 \circ \gamma \circ F(\_ \times (\pi_2 \circ \tilde{\gamma}))$ , where  $\tilde{\gamma}$  is from (8). □

**Definition 3.5.** Given a TT  $F(\_ \times X) \xrightarrow{\tilde{\gamma}} G\_ \times X$ , we define its local behavior  $LBeh(\gamma)$  to be  $F(\_ \times F^*0) \xrightarrow{\omega} G\_$  from the proof of Lemma 3.4.

In Sect. 2 we observed that a local behavior  $DA \rightarrow B$  is a coKleisli map. This is also the case in this section. In fact, the mapping  $F \mapsto F(F^*0 \times \_)$  extends to a comonad on the functor category  $[\mathbf{Sets}, \mathbf{Sets}]$ , so that any natural transformation  $F(F^*0 \times \_) \xrightarrow{\omega} G\_$  is therefore a coKleisli map from  $F$  to  $G$ . We denote their set (discrete category) by  $\mathbf{LBeh}(F, G)$ .

**Lemma 3.6.** *The operations  $LBeh$  and  $TF$  in Definitions 3.5 and 3.2 extend to functors  $LBeh : \mathbf{TT}(F, G) \rightarrow \mathbf{LBeh}(F, G)$  and  $TF : \mathbf{TT}(F, G) \rightarrow \mathbf{TF}_\uparrow(F, G)$ .  $\square$*

**Theorem 3.7.** *We have an adjunction (actually a coreflection)  $Real \dashv LBeh$ , where the realization functor for local behaviors  $Real : \mathbf{LBeh}(F, G) \rightarrow \mathbf{TT}(F, G)$  sends a local behavior  $F(\_ \times F^*0) \xrightarrow{\omega} G\_$  to a TT with Z-components*

$$F(Z \times F^*0) \xrightarrow{(\omega_Z, F\pi_2)} GZ \times FF^*0 \xrightarrow{GZ \times \alpha_0^F} GZ \times F^*0 . \quad \square$$

**Proposition 3.8.** *The functor  $W = (\mathbf{LBeh}(F, G) \xrightarrow{Real} \mathbf{TT}(F, G) \xrightarrow{TF} \mathbf{TF}_\uparrow(F, G))$  is an epimorphism.  $\square$*

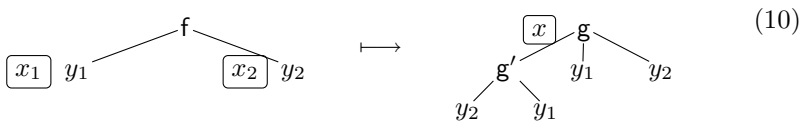
### 4 Relayering Bottom-Up Tree Transducers

In this section we will consider our most general class of bottom-up tree transformations, which can send a layer in an input tree to a truncated subtree in the output tree. For reasons of space, we must be fairly brief. We establish the same situation as in the previous section, except that we do not have to single out any condition of bottom-upness of tree functions. As we do not restrict state spaces to be finite, any tree function can arise as the behavior of a relayering bottom-up TT.

A categorical presentation of relayering TTs is obtained much like that of rebranching TTs in Sect. 3, using “placeholders-via-naturality”. We recall the notation  $F^*Z$  for the carrier of the initial  $(Z + F\_)$ -algebra. It is now important for us that the functor  $F^*$  carries a monad structure, in particular a multiplication  $\mu^F : F^*F^* \Rightarrow F^*$  that can be defined via initiality.

**Definition 4.1.** *A (relayering bottom-up) TT is a natural transformation of the form  $F(\_ \times X) \xrightarrow{\gamma} G^*\_ \times X$ . Such TTs form a category  $\mathbf{TT}(F, G)$  together with an obvious notion of morphism.*

The difference from Def. 3.1 is that we have  $G^*$  instead of  $G$  in the codomain. This corresponds to allowing terms over placeholders rather than applications of single function symbols in the right-hand sides of transition rules (7): for example,



**Definition 4.2.** A TT  $F(\_ \times X) \xrightarrow{\gamma} G^* \_ \times X$  induces its tree-function behavior  $TF(\gamma) : F^*0 \rightarrow G^*0$  by the following algebra initiality diagram.

$$\begin{array}{ccc}
 FF^*0 & \dashrightarrow & F(G^*0 \times X) \\
 \alpha_0^F \downarrow \cong & & \downarrow \gamma_{G^*0} \\
 F^*0 & \dashrightarrow \tilde{\gamma} & G^*G^*0 \times X \\
 & \searrow TF(\gamma) & \downarrow \mu_0^G \times X \\
 & & G^*0 \times X \\
 & & \downarrow \pi_1 \\
 & & G^*0
 \end{array} \tag{11}$$

For relayering TTs any tree function is bottom-up: a tree function  $l : F^*0 \rightarrow G^*0$  is realized by the TT whose  $Z$ -component is

$$F(Z \times F^*0) \xrightarrow{F\pi_2} FF^*0 \xrightarrow{\alpha_0^F} F^*0 \xrightarrow{\langle l, F^*0 \rangle} G^*0 \times F^*0 \xrightarrow{G^*! \times F^*0} G^*Z \times F^*0,$$

where  $!$  denotes the empty map  $0 \rightarrow Z$ . This realization however does not give an adjunction.

The local behavior induced by a TT  $\gamma$  is a natural transformation  $LBeh(\gamma) : F(\_ \times F^*0) \Rightarrow G^* \_$ . Such natural transformations are biKleisli maps of a distributive law of the comonad  $F \mapsto F(\_ \times F^*0)$  of the previous section over the free monad delivering monad  $F \mapsto F^*$ . We denote their set (discrete category) by  $\mathbf{LBeh}(F, G)$ .

For a realization functor for local behaviors  $Real : \mathbf{LBeh}(F, G) \rightarrow \mathbf{TT}(F, G)$  we obtain an adjunction (actually a coreflection)  $Real \dashv LBeh$ , similarly to the rebranching case.

## 5 Allowing Parameters to Vary

In Sect. 2 we saw the fundamental diagram (3) relating tree transducers, local behaviors and tree functions. In that diagram we kept the alphabets  $A, B$  fixed. In this section we shall identify additional mathematical structure that emerges by allowing the alphabets to vary. For this purpose we utilize the notion of Arrows—as introduced by Hughes [5], but described more abstractly as monoids in categories of bifunctors in [4]—and also Freyd categories (or as fibered spans).

Arrows were devised for the purpose of reconciling impure “structured computations” with purely functional computation. Commonly an Arrow  $\mathbf{A}(-, +)$  is a bifunctor  $\mathbb{C}^{op} \times \mathbb{C} \rightarrow \mathbf{Sets}$ : in this case  $\mathbf{A}(A, B)$  is the set of structured computations (of the kind designated by  $\mathbf{A}$ ) from the type  $A$  to  $B$ . Since we want to consider  $\mathbf{TT}(A, B)$  of relabeling transducers as a category of structured computation, we shall use **Cat-valued Arrows** instead: these are bifunctors  $\mathbb{C}^{op} \times \mathbb{C} \rightarrow \mathbf{Cat}$  with additional structure  $\mathbf{arr}$  and  $\ggg$  [1]. The notion of **Cat-valued Arrows** are in fact the same thing as *Freyd categories* [8] (enriched by  $\mathbf{Cat}$  in

<sup>1</sup> For the sake of brevity, we ignore here the compatibility with products which is usually given by an operation  $\mathbf{first}$ .

a suitable way): this was shown in [7]. Moreover, a **Cat**-valued Arrow—as a bifunctor  $\mathbb{C}^{\text{op}} \times \mathbb{C} \rightarrow \mathbf{Cat}$ —induces a *fibred span* via the generalized Grothendieck construction (see, e.g., [6, Ch. 9]).

In the remainder of the section we shall parameterize the diagram (3) and obtain the corresponding situation for Arrows. In this case we have  $\mathbb{C} = \mathbf{Sets}$  as the base category. We do this only for relabelings due to limited space.

The bifunctor  $\mathbf{TT}(-, +)$  is such that  $\mathbf{TT}(A, B)$  is the category of relabelings from  $A$ -trees to  $B$ -trees. It sends a morphism  $(\alpha, \beta) : (A, B) \rightarrow (C, D)$  in  $\mathbb{C}^{\text{op}} \times \mathbb{C}$ —hence  $\alpha : C \rightarrow A$  and  $\beta : B \rightarrow D$ —to the functor  $\mathbf{TT}(A, B) \rightarrow \mathbf{TT}(C, D)$  given as follows. On objects:

$$(A \times FX \xrightarrow{c} B \times X) \mapsto (C \times FX \xrightarrow{\alpha \times FX} A \times FX \xrightarrow{c} B \times FX \xrightarrow{\beta \times FX} D \times FX)$$

and on morphisms it is the identity.

Interestingly, there is also a monoid structure  $\mathbf{TT} \otimes \mathbf{TT} \xrightarrow{\cong} \mathbf{TT} \overset{\text{arr}}{I}$  on the bifunctor  $\mathbf{TT}$ —this makes  $\mathbf{TT}$  an Arrow (see [4]). We shall describe it a bit more concretely. For TTs  $A \times FX \xrightarrow{c} C \times X$  and  $C \times FY \xrightarrow{d} B \times Y$  with matching output/input, their composition  $c \ggg d$  has  $X \times Y$  as its state space:

$$A \times \underline{F(X \times Y)} \xrightarrow{\langle F\pi_1, F\pi_2 \rangle} \underline{A \times FX} \times FY \xrightarrow{c} \underline{C} \times X \times \underline{FY} \xrightarrow{d} B \times X \times Y .$$

The operation *arr* for  $\mathbf{TT}$  carries a morphism  $A \xrightarrow{f} B$  in  $\mathbb{C}$  to a TT with a trivial state space 1: namely  $A \times F1 \xrightarrow{\pi_1} A \xrightarrow{f} B \xrightarrow{\cong} B \times 1$ . It is easy to check that *arr* and  $\ggg$  satisfy the appropriate naturality and monoid equations.

Just like  $\mathbf{TT}(-, +)$  carries the structure of an Arrow we can identify similar structure on  $\mathbf{LBeh}(-, +)$ ,  $\mathbf{TF}(-, +)$  and  $\mathbf{TF}_\uparrow(-, +)$ . It then turns out that the diagram (3), but then without the fixed alphabets, also exists in parameterized form, even with preservation of this Arrow structure. For example, the behavior-realization adjunction is now described as an adjunction between Arrows.

**Theorem 5.1.** *We have the following situation in the 2-category **Arrow**.*

$$\begin{array}{ccc}
 \mathbf{TT}(-, +) & \xrightarrow{TF} & \mathbf{TF}_\uparrow(-, +) \hookrightarrow \mathbf{TF}(-, +) \\
 \text{Real} \left\{ \begin{array}{l} \uparrow \\ \downarrow \end{array} \right. \mathbf{LBeh} & & \\
 \mathbf{LBeh}(-, +) & \xrightarrow{W} & 
 \end{array} \tag{12}$$

□

## 6 Conclusions and Future Work

We have given a categorical account of three classes of bottom-up tree transformations. Notably, we have generalized traditional signatures to functors and replaced traditional descriptions of TTs based on placeholder notation with natural transformations, winning simplicity and clarity. In future work, we will

elaborate on our basic picture in a form where, in addition to “extensional” tree functions, we also have “intensional” tree functions, capable of tracking which node in an input tree goes where in the output tree. And we will also include top-down computations, using the theory of containers, as well as bottom-up and top-down computations with look-ahead.

*Acknowledgement.* T. Uustalu was partially supported by the Estonian Science Foundation grants No. 5567 and 6940.

## References

1. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications. Book draft (2005)
2. Engelfriet, J.: Bottom-up and top-down tree transformations—a comparison. *Math. Syst. Theory* 9(3), 198–231 (1975)
3. Goguen, J.: Minimal realization of machines in closed categories. *Bull. Amer. Math. Soc.* 78(5), 777–783 (1972)
4. Heunen, C., Jacobs, B.: Arrows, like monads, are monoids. In: Brookes, S., Mislove, M. (eds.) *Proc. of 22nd Conf. on Math. Found. of Program. Semantics, MFPS-XXII*. *Electr. Notes in Theor. Comput. Sci.*, vol. 158, pp. 219–236. Elsevier, Amsterdam (2006)
5. Hughes, J.: Generalising monads to arrows. *Sci. of Comput. Program.* 37(1–3), 67–111 (2000)
6. Jacobs, B.: *Categorical Logic and Type Theory*. North-Holland, Amsterdam (1999)
7. Jacobs, B., Hasuo, I.: Freyd is Kleisli, for arrows. In: McBride, C., Uustalu, T. (eds.) *Proc. of Wksh. on Mathematically Structured Functional Programming, MSFP '06*, *Electron. Wkshs. in Comput. Sci.*, BCS (2006)
8. Power, J., Robinson, E.: Premonoidal categories and notions of computation. *Math. Struct. in Comput. Sci.* 7(5), 453–468 (1997)
9. Rosebrugh, R.D., Sabadini, N., Walters, R.F.C.: Minimal realization in bicategories of automata. *Math. Struct. in Comput. Sci.* 8(2), 93–116 (1998)
10. Rounds, W.C.: Mappings and grammars on trees. *Math. Syst. Theory* 4(3), 257–287 (1970)
11. Thatcher, J.W.: Generalized sequential machine maps. *J. Comput. Syst. Sci.* 4(4), 339–367 (1970)
12. Uustalu, T., Vene, V.: Comonadic functional attribute evaluation. In: van Eekelen, M. (ed.) *Trends in Functional Programming 6*, Intellect, pp. 145–162 (2007)
13. Uustalu, T., Vene, V.: The essence of dataflow programming. In: Horváth, Z. (ed.) *CEFP 2005*. *LNCS*, vol. 4164, pp. 135–167. Springer, Heidelberg (2006)
14. Uustalu, T., Vene, V.: The dual of substitution is redecoration. In: Hammond, K., Curtis, S. (eds.) *Trends in Functional Programming 3*, Intellect, pp. 99–110 (2002)

# Holographic Algorithms: The Power of Dimensionality Resolved

Jin-Yi Cai<sup>1,\*</sup> and Pinyan Lu<sup>2,\*\*</sup>

<sup>1</sup> Computer Sciences Department, University of Wisconsin  
Madison, WI 53706, USA  
jyc@cs.wisc.edu

<sup>2</sup> Department of Computer Science and Technology, Tsinghua University  
Beijing, 100084, P.R. China  
lpy@mails.tsinghua.edu.cn

**Abstract.** Valiant's theory of holographic algorithms is a novel methodology to achieve exponential speed-ups in computation. A fundamental parameter in holographic algorithms is the dimension of the linear basis vectors. We completely resolve the problem of the power of higher dimensional bases. We prove that 2-dimensional bases are universal for holographic algorithms.

## 1 Introduction

Complexity theory has learned a great deal about the nature of efficient computation. However if the ultimate goal is to gain a fundamental understanding such as what differentiates polynomial time from exponential time, we are still a way off. In fact, in the last 20 years, the most spectacular advances in the field have come from discovering new and surprising ways to do efficient computations. The theory of holographic algorithms introduced recently by Valiant [18] is one such new methodology which gives polynomial time algorithms to some problems which seem to require exponential time.

To describe this theory requires some background. At the top level it is a method to represent computational information in a superposition of linear vectors, somewhat analogous to quantum computing. This information is manipulated algebraically, but in a purely classical way. Then via a beautiful theorem called the Holant Theorem [18], which expresses essentially an invariance of tensor contraction under basis transformations [2], this computation is reduced to the computation of perfect matchings in planar graphs. It so happens that counting perfect matchings for planar graphs is computable in polynomial time by the elegant FKT method [11][2][15]. Thus we obtain a polynomial time algorithm. The whole exercise can be thought of as an elaborate scheme to introduce a custom made process of exponential cancellations. The end result is a polynomial time evaluation of an exponential sum which expresses the desired computation.

---

\* Supported by NSF CCR-0511679.

\*\* Supported by the National Natural Science Foundation of China Grant 60553001 and the National Basic Research Program of China Grant 2007CB807900,2007CB807901.



On a more technical level, there are two main ingredients in the design of a holographic algorithm. First, a collection of planar matchgates. Second, a choice of linear basis vectors, through which the computation is expressed and interpreted. Typically there are two basis vectors  $n$  and  $p$  in dimension 2, which represent the bit values 0 and 1 respectively, and their tensor product will represent a combination of 0-1 bits. It is the superpositions of these vectors in the tensor product space that are manipulated by a holographic algorithm in the computation. This superposition gives rise to exponential sized aggregates with which massive cancellations take place. In this sense holographic algorithms are more akin to quantum algorithms than to classical algorithms in their design and operation.

No polynomial time algorithms were known previously for any of the problems in [18,21,22], and some minor variations are NP-hard. These problems may also appear quite restricted. Here is a case in point. Valiant showed [21] that the problem  $\#_7\text{Pl-Rtw-Mon-3CNF}$  is solvable in P by this method. This problem is a restrictive Satisfiability counting problem. Given a planar read-twice monotone 3CNF formula, it counts the number of satisfying assignments, modulo 7. However, it is known that even for this restricted class of Boolean formulae, the counting problem without the modulo 7 is  $\#P$ -complete. Also, the counting problem modulo 2 (denoted as  $\#_2\text{Pl-Rtw-Mon-3CNF}$ ) is  $\oplus P$ -complete (thus NP-hard by randomized reductions). The ultimate power of this theory is unclear.

It is then natural to ask, whether holographic algorithms will bring about a collapse of complexity classes. Regarding conjectures such as  $P \neq NP$  undogmatically, it is incumbent for us to gain a systematic understanding of the capabilities of holographic algorithms. This brings us closer to the fundamental reason why these algorithms are fascinating—its implication for complexity theory. The fact that some of these problems such as  $\#_7\text{Pl-Rtw-Mon-3CNF}$  might appear a little contrived is beside the point. When potential algorithmic approaches to P vs. NP were surveyed, these algorithms were not part of the repertoire; presumably the same “intuition” for  $P \neq NP$  would have applied equally to  $\#_7\text{Pl-Rtw-Mon-3CNF}$  and to  $\#_2\text{Pl-Rtw-Mon-3CNF}$ .

In holographic algorithms, since the underlying computation is ultimately reduced to perfect matchings, the linear basis vectors which express the computation are necessarily of dimension  $2^k$ , for some integer  $k$ . This  $k$  is called the size of the basis. Most holographic algorithms so far [18,21,22] use bases of size 1. Surprisingly Valiant’s algorithm for  $\#_7\text{Pl-Rtw-Mon-3CNF}$  used a basis of size 2. Utilizing bases of a higher dimension has always been a theoretical possibility, which may further extend the reach of holographic algorithms. Valiant’s algorithm makes it realistic.

It turns out that for  $\#_7\text{Pl-Rtw-Mon-3CNF}$  one can design another holographic algorithm with a basis of size 1 [4]. Subsequently we have proved [6] the surprising result that any basis of size 2 can be replaced by a suitable basis of size 1 in a holographic algorithm. In this paper we completely resolve the problem of whether bases of higher dimensions are more powerful. *They are not.*

Our starting point is a theorem from [6] concerning degenerate tensors of matchgates. For bases of size 2 we were able to find explicit constructions of certain gadgets from scratch. But this approach encountered major difficulties for arbitrary size  $k$ . The underlying reason for this is that for larger matchgates there is a set of exponential sized algebraic constraints called matchgate identities [17][13] which control their realizability. This additional constraint is absent for small matchgates. The difficulty is finally overcome by deriving a tensor theoretic decomposition. This reveals an internal structure for non-degenerate matchgate tensors. We discover that for any basis of size  $k$ , except in a degenerate case, there is an embedded basis of size 1. To overcome the difficulty of realizability, we make use of the given matchgates on a basis of size  $k$ , and “fold” these matchgates onto themselves to get new matchgates on the embedded basis of size 1. These give geometric realizations, by planar graphs, of those tensors in the decomposition which were defined purely algebraically. Thus we are able to completely bypass matchgate identities here. In the process, we gain a substantial understanding of the structure of a general holographic algorithm on a basis of size  $k$ .

This paper is organized as follows. In Section 2, we give a brief summary of background information. In Section 3, we give a structural theorem for valid bases, the tensor theoretic decomposition, and prove two key theorems for the realizability of generators. In Section 4, we prove a realizability theorem for recognizers. This leads to the main theorem.

## 2 Background

Let  $G = (V, E, W)$  be a weighted undirected planar graph. A *generator matchgate*  $\Gamma$  is a tuple  $(G, X)$  where  $X \subseteq V$  is a set of external *output* nodes. A *recognizer matchgate*  $\Gamma'$  is a tuple  $(G, Y)$  where  $Y \subseteq V$  is a set of external *input* nodes. The external nodes are ordered counter-clockwise on the external face.  $\Gamma$  (or  $\Gamma'$ ) is called an odd (resp. even) matchgate if it has an odd (resp. even) number of nodes.

Each matchgate is assigned a *signature* tensor. A generator  $\Gamma$  with  $n$  output nodes is assigned a contravariant tensor  $\mathbf{G}$  of type  $\binom{n}{0}$ . Under the standard basis, it takes the form  $\underline{G}$  with  $2^n$  entries, where

$$\underline{G}^{i_1 i_2 \dots i_n} = \text{PerfMatch}(G - Z).$$

Here  $\text{PerfMatch}$  is the sum of all weighted perfect matchings, and  $Z$  is the subset of the output nodes having the characteristic sequence  $\chi_Z = i_1 i_2 \dots i_n$ .  $\underline{G}$  is called the standard signature of the generator  $\Gamma$ . We can view  $\underline{G}$  as a column vector (whose entries are ordered lexicographically according to  $\chi_Z$ ).

Similarly a recognizer  $\Gamma' = (G', Y)$  with  $n$  input nodes is assigned a covariant tensor  $\mathbf{R}$  of type  $\binom{0}{n}$ .

Because of the parity constraint of perfect matchings, half of all entries of a standard signature  $\underline{G}$  (or  $\underline{R}$ ) are zero. Therefore, we can use a tensor in  $V_0^{n-1}$  (or

$V_{n-1}^0$ ) to represent all the information contained in  $\underline{G}$  (or  $\underline{R}$ ). More precisely, we have the following definition (we only need for the generators).

**Definition 1.** *If a generator matchgate  $\Gamma$  with arity  $n$  is even (resp. odd), a condensed standard signature  $\underline{G}$  of  $\Gamma$  is a tensor in  $V_0^{n-1}$ , and  $\underline{G}^\alpha = \underline{G}^{\alpha b}$  (resp.  $\underline{G}^\alpha = \underline{G}^{\alpha \bar{b}}$ ), where  $\underline{G}$  is the standard signature of  $\Gamma$ ,  $\alpha \in \{0, 1\}^{n-1}$  and  $b = \oplus \alpha$  is the sum of the bits of  $\alpha$  mod 2, i.e., the parity of the Hamming weight of  $\alpha$ .*

A basis  $T$  contains 2 vectors  $(t_0, t_1)$  (also denoted as  $(n, p)$ ), each of them has dimension  $2^k$  (size  $k$ ). We use the following notation:  $T = (t_i^\alpha) = [n^\alpha, p^\alpha]$ , where  $i \in \{0, 1\}$  and  $\alpha \in \{0, 1\}^k$ . We follow the convention that upper index  $\alpha$  is for row and lower index  $i$  is for column (see [8]). We assume  $\text{rank}(T) = 2$  in the following discussion because a basis of  $\text{rank}(T) \leq 1$  is useless. Under a basis  $T$ , we can talk about non-standard signatures (or simply signatures).

**Definition 2.** *The contravariant tensor  $\mathbf{G}$  of a generator  $\Gamma$  has signature  $G$  under basis  $T$  iff  $\underline{G} = T^{\otimes n} \mathbf{G}$  is the standard signature of the generator  $\Gamma$ .*

**Definition 3.** *The covariant tensor  $\mathbf{R}$  of a recognizer  $\Gamma'$  has signature  $R$  under basis  $T$  iff  $R = \underline{R} T^{\otimes n}$ , where  $\underline{R}$  is the standard signature of the recognizer  $\Gamma'$ .*

We have

$$\underline{G}^{\alpha_1 \alpha_2 \dots \alpha_n} = \sum_{i_1, i_2, \dots, i_n \in \{0, 1\}} G^{i_1 i_2 \dots i_n} t_{i_1}^{\alpha_1} t_{i_2}^{\alpha_2} \dots t_{i_n}^{\alpha_n} \tag{1}$$

$$R_{i_1 i_2 \dots i_n} = \sum_{\alpha_1, \alpha_2, \dots, \alpha_n \in \{0, 1\}^k} \underline{R}_{\alpha_1 \alpha_2 \dots \alpha_n} t_{i_1}^{\alpha_1} t_{i_2}^{\alpha_2} \dots t_{i_n}^{\alpha_n} \tag{2}$$

**Definition 4.** *A contravariant tensor  $\mathbf{G} \in V_0^n$  (resp. a covariant tensor  $\mathbf{R} \in V_n^0$ ) is realizable on a basis  $T$  iff there exists a generator  $\Gamma$  (resp. a recognizer  $\Gamma'$ ) such that  $G$  (resp.  $R$ ) is the signature of  $\Gamma$  (resp.  $\Gamma'$ ) under basis  $T$ .*

For a string  $\alpha \in \{0, 1\}^n$ , we use the notation  $\text{wt}(\alpha)$  to denote its Hamming weight. A signature  $G$  or  $R$  on index  $\alpha = \alpha_1 \alpha_2 \dots \alpha_n$ , where each  $\alpha_i \in \{0, 1\}^k$ , is *symmetric* iff the value of  $G^\alpha$  or  $R_\alpha$  only depends on the number of  $k$ -bit patterns of  $\alpha_i$ , i.e., it is symmetric under permutations of the blocks  $\alpha_i$ . For  $k = 1$  it only depends on the Hamming weight  $\text{wt}(\alpha)$  of its index  $\alpha$ . For  $k = 1$ , we can denote a symmetric signature by the notation  $[z_0, z_1, \dots, z_n]$ , where  $i$  is the Hamming weight, and  $z_i$  is the value of the signature for an index of  $\text{wt}(\alpha) = i$ . We note that  $k = 1$  always for signatures other than standard signatures.

A *matchgrid*  $\Omega = (A, B, C)$  is a weighted planar graph consisting of a disjoint union of: a set of  $g$  generators  $A = (A_1, \dots, A_g)$ , a set of  $r$  recognizers  $B = (B_1, \dots, B_r)$ , and a set of  $f$  connecting edges  $C = (C_1, \dots, C_f)$ , where each  $C_i$  edge has weight 1 and joins an output node of a generator with a input node of a recognizer, so that every input and output node in every constituent matchgate has exactly one such incident connecting edge.

Let  $G(A_i, T)$  be the signature of generator  $A_i$  under the basis  $T$  and  $R(B_j, T)$  be the signature of recognizer  $B_j$  under the basis  $T$ . And Let  $G = \bigotimes_{i=1}^g G(A_i, T)$

and  $R = \bigotimes_{j=1}^r R(B_j, T)$ . Then  $\text{Holant}(\Omega)$  is defined to be the contraction of these two product tensors, where the corresponding indices match up according to the  $f$  connecting edges in  $C$ . We note that for a holographic algorithm to use a basis of size  $k > 1$ , each matchgate of arity  $n$  in the matchgrid has  $kn$  external nodes, grouped in blocks of  $k$  nodes each. These  $k$  nodes are connected in a block-wise fashion between matchgates, where the combinations of tensor products of the  $2^k$ -dimensional basis vectors are interpreted as truth values.

**Theorem 1 (Valiant).** *For any matchgrid  $\Omega$  over any basis  $T$ , let  $G$  be its underlying weighted graph, then*

$$\text{Holant}(\Omega) = \text{PerfMatch}(G).$$

There is a subtlety for the universal bases collapse theorem. It turns out that if we only focus on the recognizers, bases of size  $k > 1$  are in fact provably more powerful than bases of size 1. It is only in the context of simultaneous realizability of both generators and recognizers that we are able to achieve this universal collapse. The first crucial insight is to isolate certain degenerate bases.

**Definition 5.** *A basis  $T$  is degenerate iff  $t^\alpha = (t_0^\alpha, t_1^\alpha) = 0$  for all  $\text{wt}(\alpha)$  even (or for all  $\text{wt}(\alpha)$  odd).*

**Definition 6.** *A generator tensor  $G \in V_0^n$  ( $\dim(V) = 2$ ) is degenerate iff it has the following form (where  $G_i \in V$  is a arity 1 tensor):*

$$G = G_1 \otimes G_2 \otimes \cdots \otimes G_n. \tag{3}$$

Degenerate generators can be completely decoupled. A holographic algorithm that uses only degenerate generators has no connections between its various components and hence is essentially trivial.

In [6], we proved the following theorem. The proof uses matchgate identities.

**Theorem 2.** *If a basis  $T$  is degenerate and  $\text{rank}(T) = 2$ , then every generator  $G \in V_0^n$  realizable on the basis  $T$  is degenerate.*

### 3 Valid Bases

**Definition 7.** *A basis  $T$  is valid iff there exists some non-degenerate generator realizable on  $T$ .*

Our starting point is a careful study of high dimensional valid bases.

**Corollary 1.** *A valid basis is non-degenerate.*

**Theorem 3.** *For every valid basis  $T = [n, p]$ ,  $(n^\alpha, p^\alpha)$  and  $(n^\beta, p^\beta)$  are linearly dependent, for all  $\text{wt}(\alpha), \text{wt}(\beta)$  having the same parity.*

**Proof:** Since  $T = [n, p]$  is valid, by definition, there exists a non-degenerate generator  $G$  which is realizable on  $T$ . From Corollary [11](#), we know that  $T = [n, p]$  is non-degenerate.

Let  $\alpha_0, \beta_0$  be two arbitrary indices of even weight and  $\alpha_1, \beta_1$  be two arbitrary indices of odd weight. Let  $T_0 = \left[ \begin{pmatrix} n^{\alpha_0} \\ n^{\beta_0} \end{pmatrix}, \begin{pmatrix} p^{\alpha_0} \\ p^{\beta_0} \end{pmatrix} \right]$  and  $T_1 = \left[ \begin{pmatrix} n^{\alpha_1} \\ n^{\beta_1} \end{pmatrix}, \begin{pmatrix} p^{\alpha_1} \\ p^{\beta_1} \end{pmatrix} \right]$ . Then we need to prove  $\det(T_0) = \det(T_1) = 0$ .

According to the parity of the arity  $n$  and the parity of the matchgate realizing  $G$ , we have 4 cases:

**Case 1: even  $n$  and odd matchgate**

From the parity constraint, we have  $T_0^{\otimes n}G = 0$  and  $T_1^{\otimes n}G = 0$ . Since  $G \neq 0$  (i.e.,  $G$  is not identically 0), we have  $\det(T_0) = \det(T_1) = 0$ . Note that  $\det(T^{\otimes n}) = (\det(T))^{n2^{n-1}}$ .

**Case 2: odd  $n$  and odd matchgate**

From the parity constraint, we have  $T_0^{\otimes n}G = 0$ . Since  $G \neq 0$ , we have  $\det(T_0) = 0$ . Since the basis is non-degenerate, from the definition, there exists a  $\alpha$  such that  $\text{wt}(\alpha)$  is even and  $(n^\alpha, p^\alpha) \neq (0, 0)$ .

From the parity constraint, for all  $t \in [n] = \{1, \dots, n\}$ , we have

$$(T_1^{\otimes(t-1)} \otimes (n^\alpha, p^\alpha) \otimes T_1^{\otimes(n-t)})G = 0. \tag{4}$$

Let  $G_t$  be the tensor of type  $V_0^{n-1}$  defined by

$$G_t^{i_1 i_2 \dots i_{n-1}} = n^\alpha G^{i_1 i_2 \dots i_{t-1} 0 i_t i_{t+1} \dots i_{n-1}} + p^\alpha G^{i_1 i_2 \dots i_{t-1} 1 i_t i_{t+1} \dots i_{n-1}},$$

where  $i_1, i_2, \dots, i_{n-1} = 0, 1$ . Then equation [\(4\)](#) translates to  $T_1^{\otimes(n-1)}G_t = 0$ .

If  $\forall t \in [n]$  we have  $G_t \equiv 0$ , then we claim  $G$  is symmetric and degenerate. To see this, first suppose  $p^\alpha \neq 0$ . Then for all  $i_1, i_2, \dots, i_n = 0, 1$ ,  $G^{i_1 i_2 \dots i_n} = G^{00 \dots 0}(-n^\alpha/p^\alpha)^{\text{wt}(i_1 i_2 \dots i_n)}$ . This is clearly symmetric, and degenerate by [\(3\)](#). The proof is similar if  $n^\alpha \neq 0$ . Since by assumption  $(n^\alpha, p^\alpha) \neq (0, 0)$ , it follows that  $G$  is degenerate. This is a contradiction.

Therefore there exists some  $t \in [n]$  such that  $G_t \neq 0$ . Then from  $T_1^{\otimes(n-1)}G_t = 0$ , we have  $\det(T_1) = 0$ .

**Case 3: odd  $n$  and even matchgate**

This is similar to Case 2. We apply the argument for  $T_0$  to  $T_1$ , and apply the argument for  $T_1$  to  $T_0$ .

**Case 4: even  $n$  and even matchgate**

This case is also similar to Case 2 and Case 3. We simply apply the same argument for  $T_1$  as in Case 2 and the same argument for  $T_0$  as in Case 3. □

From this theorem, we know that for any valid basis  $T = [n^\alpha, p^\alpha]$  (where  $\alpha \in \{0, 1\}^k$ ), there exist non-zero vectors  $(n^{\alpha_0}, p^{\alpha_0})$ , and  $(n^{\alpha_1}, p^{\alpha_1})$ , where  $\alpha_0, \alpha_1 \in \{0, 1\}^k$ , and  $\text{wt}(\alpha_0)$  is even and  $\text{wt}(\alpha_1)$  is odd, such that every other  $(n^\alpha, p^\alpha)$  is a scalar multiple of one of these two vectors (the one with the same parity). More precisely, we define  $\hat{n}^b = n^{\alpha^b}$  and  $\hat{p}^b = p^{\alpha^b}$  for  $b = 0, 1$ , then there exist  $\lambda^\alpha$  for all  $\alpha \in \{0, 1\}^k$ , such that  $(n^\alpha, p^\alpha) = \lambda^\alpha(\hat{n}^{\oplus\alpha}, \hat{p}^{\oplus\alpha})$ , where  $\oplus\alpha$  is the parity of  $\text{wt}(\alpha)$ .

Note that  $(\hat{n}^0, \hat{p}^0), (\hat{n}^1, \hat{p}^1)$  are linearly independent, otherwise  $\text{rank}(T) < 2$ . Therefore each is determined up to a scalar multiplier. This justifies the following

**Definition 8.** We call  $\hat{T} = \left[ \begin{pmatrix} \hat{n}^0 \\ \hat{n}^1 \end{pmatrix}, \begin{pmatrix} \hat{p}^0 \\ \hat{p}^1 \end{pmatrix} \right]$  an embedded size 1 basis of  $T$ .

Now suppose a non-degenerate generator  $G$  is realizable on a valid basis  $T = [n^\alpha, p^\alpha]$ , (where  $\alpha \in \{0, 1\}^k$ ), and  $\hat{T} = (\hat{t}_i^\alpha)$  is an embedded size 1 basis of  $T$ .

Substituting  $(t_0^\alpha, t_1^\alpha) = \lambda^\alpha (\hat{t}_0^{\oplus\alpha}, \hat{t}_1^{\oplus\alpha})$  in (4), we have

$$\begin{aligned} \underline{G}^{\alpha_1\alpha_2\cdots\alpha_n} &= \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} G^{i_1 i_2 \cdots i_n} t_{i_1}^{\alpha_1} t_{i_2}^{\alpha_2} \dots t_{i_n}^{\alpha_n} \\ &= \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} G^{i_1 i_2 \cdots i_n} \lambda^{\alpha_1} \hat{t}_{i_1}^{\oplus\alpha_1} \lambda^{\alpha_2} \hat{t}_{i_2}^{\oplus\alpha_2} \dots \lambda^{\alpha_n} \hat{t}_{i_n}^{\oplus\alpha_n} \\ &= \lambda^{\alpha_1} \lambda^{\alpha_2} \dots \lambda^{\alpha_n} \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} G^{i_1 i_2 \cdots i_n} \hat{t}_{i_1}^{\oplus\alpha_1} \hat{t}_{i_2}^{\oplus\alpha_2} \dots \hat{t}_{i_n}^{\oplus\alpha_n}. \end{aligned}$$

We define a tensor  $\hat{G} \in V_0^n$  as follows: For  $j_1, j_2, \dots, j_n = 0, 1$ ,

$$\hat{G}^{j_1 j_2 \cdots j_n} = \sum_{i_1, i_2, \dots, i_n \in \{0,1\}} G^{i_1 i_2 \cdots i_n} \hat{t}_{i_1}^{j_1} \hat{t}_{i_2}^{j_2} \dots \hat{t}_{i_n}^{j_n}. \tag{5}$$

Then we have

$$\underline{G}^{\alpha_1\alpha_2\cdots\alpha_n} = \lambda^{\alpha_1} \lambda^{\alpha_2} \dots \lambda^{\alpha_n} \hat{G}^{\oplus\alpha_1 \oplus\alpha_2 \cdots \oplus\alpha_n}. \tag{6}$$

The decomposition (6) is pregnant with structural information (see discussion in 7). Starting with any non-degenerate  $G$  which is realizable on a valid basis  $T$ , we defined its embedded size 1 basis  $\hat{T}$ ,  $(\lambda^\alpha)$  and  $\hat{G}$  by (5). But we note that (5) and (6) are satisfied for every generator (we only need one non-degenerate  $G$  to establish  $\hat{T}$ ). Then regarding (6) we have the following key theorems:

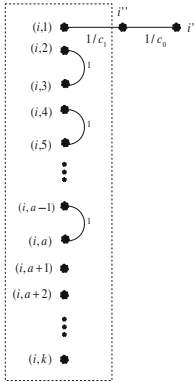
**Theorem 4.**  $(\lambda^\alpha)$  (where  $\alpha \in \{0, 1\}^k$ ) is a condensed signature of some generator matchgate with arity  $k + 1$ .

**Theorem 5.**  $\hat{G}$  is a standard signature of some generator matchgate of arity  $n$ .

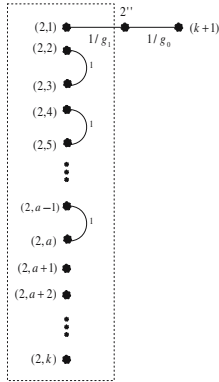
The proofs of Theorems 4 and 5 are both constructive. We make one more definition. Since the basis  $T$  is non-degenerate, there exist  $\beta_0$  and  $\beta_1$ , such that  $\text{wt}(\beta_0)$  is even,  $\text{wt}(\beta_1)$  is odd, and  $\lambda^{\beta_0} \lambda^{\beta_1} \neq 0$ . We also assume  $\beta_0$  and  $\beta_1$  is such a pair with minimum Hamming distance. To simplify notations in the following proof, we assume  $\beta_0 = 00 \cdots 0$  and  $\beta_1 = 11 \cdots 100 \cdots 0$  (where there are  $a$  1s,  $a$  is odd). This simplifying assumption is without loss of generality; we omit this justification here and it can be found in the full paper 7.

Let  $c_0 = \lambda^{\beta_0} = \lambda^{00 \cdots 000 \cdots 0}$  and  $c_1 = \lambda^{\beta_1} = \lambda^{11 \cdots 100 \cdots 0}$ . In this setting, for any pattern  $\gamma$  strictly between  $\beta_0$  and  $\beta_1$  (if any), if  $\alpha_r = \gamma$  for some  $r \in [n]$ , then by (6)

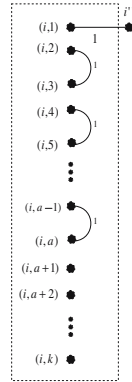
$$\underline{G}^{\alpha_1\alpha_2\cdots\alpha_n} = 0. \tag{7}$$



**Fig. 1.** Modify the  $i$ -th block of  $\Gamma$  to get the  $i$ -th external node of  $\widehat{\Gamma}$



**Fig. 2.** Modify the second block of  $\Gamma$  to get the  $(k + 1)$ -th external node of  $\Gamma_\lambda$



**Fig. 3.** Modify the  $i$ -th block of  $\Gamma$  when  $j_i = 1$ . All the nodes are viewed as internal in  $\Gamma_\lambda$ .

Since  $G$  is realizable on  $T$ ,  $\underline{G}$  is the standard signature of some matchgate  $\Gamma$  with arity  $nk$ . For convenience, we label its  $((i - 1)k + j)$ -th external node by a pair of integers  $(i, j)$ , where  $i \in [n], j \in [k]$ .

**Proof of Theorem 5:** For every  $i \in [n]$ , do the following modifications to the  $k$  nodes  $(i, j)$  of the  $i$ -th block of external nodes in  $\Gamma$ , where  $j \in [k]$  (see Fig. 1):

- Connect  $(i, l)$  with  $(i, l + 1)$  by an edge of weight 1, for  $l = 2, 4, \dots, a - 1$ .
- Add two new nodes  $i'$  and  $i''$ .
- Connect  $(i, 1)$  and  $i''$  by an edge of weight  $1/c_1$ .
- Connect  $i''$  and  $i'$  by an edge of weight  $1/c_0$ .

After all these modifications, viewing the  $n$  nodes  $i'$  (one node stemming from each block,  $i \in [n]$ ) as external nodes and all other nodes as internal nodes, we have a matchgate  $\widehat{\Gamma}$  with arity  $n$ . Now we prove that  $\widehat{G}$  is the standard signature of this matchgate  $\widehat{\Gamma}$ .

Denote the standard signature of  $\widehat{\Gamma}$  temporarily as  $(\widehat{\Gamma}^{j_1 j_2 \dots j_n})$ . For an arbitrary pattern  $j_1 j_2 \dots j_n \in \{0, 1\}^n$ , we consider the value  $\widehat{\Gamma}^{j_1 j_2 \dots j_n}$ . For  $r \in [n]$ , there are two cases:

- Case 1:  $j_r = 0$ . In this case, we keep the external node  $r'$ . Any perfect matching will take the edge  $(r'', r')$ , this contributes a factor of  $1/c_0$ . As a result, the node  $(r, 1)$  must match with some node in the original  $\Gamma$ . And from (7), the only possible non-zero pattern of this block of  $\underline{G}$  is  $\beta_0 = 00 \dots 0$ . (This means that the perfect matchings will not take any of the new weight 1 edges.)
- Case 2:  $j_r = 1$ . In this case, we remove the external node  $r'$ . Any perfect matching will take the edge between  $(r, 1)$  and  $r''$ , this contributes a factor of  $1/c_1$ . As a result, the node  $(r, 1)$  will be removed from the original  $\Gamma$ . And

from (7), the only possible non-zero pattern of this block of  $\underline{G}$  is  $\beta_1$ . (This means that the perfect matchings will take all of the new weight 1 edges.)

To sum up,

$$\widehat{\Gamma}^{j_1 j_2 \dots j_n} = \frac{1}{c_{j_1}} \frac{1}{c_{j_2}} \dots \frac{1}{c_{j_n}} \underline{G}^{\beta_{j_1} \beta_{j_2} \dots \beta_{j_n}}.$$

Together with (6), we know this is exactly  $\widehat{G}$ . This completes the proof.  $\square$

Before we prove Theorem 4, we have the following claim. The proof is omitted here and can be found in the full paper [7].

**Claim 1.** For any standard signature with more than one non-zero entries, there exist two non-zero entries  $G^\alpha$  and  $G^\beta$  such that the Hamming distance between  $\alpha$  and  $\beta$  is 2.

**Proof of Theorem 4:** Here we start with a non-degenerate  $G$ . By Claim 3, for notational simplicity we assume  $G_0 = \widehat{G}^{00j_3j_4\dots j_n} \neq 0$  and  $G_1 = \widehat{G}^{11j_3j_4\dots j_n} \neq 0$ . Other cases can be proved similarly. We are given the planar matchgate  $\Gamma$  with standard signature  $\underline{G}$ . We carry out the following transformations of  $\Gamma$ :

- Do nothing to the first block. However, for convenience, we rename the first  $k$  nodes as  $1', 2', \dots, k'$ .
- Change the second block as in Figure 2, where  $g_0 = G_0 \lambda^{\beta_0} \lambda^{\beta_{j_3}} \dots \lambda^{\beta_{j_n}}$  and  $g_1 = G_1 \lambda^{\beta_1} \lambda^{\beta_{j_3}} \dots \lambda^{\beta_{j_n}}$ . Note that  $g_0, g_1 \neq 0$ . It has a new external node  $(k+1)'$ .
- For  $i \geq 3$  and  $j_i = 0$ , do nothing to the  $i$ -th block.
- For  $i \geq 3$  and  $j_i = 1$ , change the  $i$ -th block as in Figure 3.

After all these changes, we will consider the  $k+1$  nodes  $i'$  (where  $i \in [k+1]$ , the first  $k$  nodes all stem from the first block, and  $(k+1)'$  stems from the second block) as the new external nodes and all other nodes as internal nodes. In this way we obtain a planar matchgate  $\Gamma_\lambda$  with arity  $k+1$ . Now we prove that  $\lambda^\alpha$  is the condensed standard signature of  $\Gamma_\lambda$ .

First we show that  $\Gamma_\lambda$  is an even matchgate. Let  $x$  be the number of nodes in  $\Gamma$  and  $y = \text{wt}(j_3 j_4 \dots j_n)$ . Since

$$\underline{G}^{\beta_0 \beta_0 \beta_{j_3} \beta_{j_4} \dots \beta_{j_n}} = \lambda^{\beta_0} \lambda^{\beta_0} \lambda^{\beta_{j_3}} \lambda^{\beta_{j_4}} \dots \lambda^{\beta_{j_n}} \widehat{G}^{00j_3\dots j_n} \neq 0,$$

we know  $x - ya$  is even. Given that  $a$  is odd, we can count mod 2, and get  $x + y + 2 \equiv x - ya \equiv 0 \pmod{2}$ . Since  $x + y + 2$  is exactly the number of nodes in  $\Gamma_\lambda$ , we know  $\Gamma_\lambda$  is an even matchgate.

For  $\alpha \in \{0, 1\}^k$  and  $\text{wt}(\alpha)$  is even, we consider  $\Gamma_\lambda^{\alpha 0}$  at the  $(k+1)$ -bit pattern  $\alpha 0$ . Consider each block in turn in  $\Gamma$ . The first block clearly should be given the  $k$ -bit pattern  $\alpha$ . The only possible non-zero value concerning the second block is to take the edge  $(2'', (k+1)')$  with weight  $1/g_0$ , and assign the all-0 pattern  $\beta_0$  to  $(2, 1), (2, 2), \dots, (2, k)$ . This follows from (7). Similarly for the  $i$ -th block, where  $i \geq 3$ , we must assign the pattern  $\beta_{j_i}$ . Hence, applying (6) we get,

$$\Gamma_\lambda^{\alpha 0} = \frac{1}{g_0} \underline{G}^{\alpha \beta_0 \beta_{j_3} \beta_{j_4} \dots \beta_{j_n}} = \frac{1}{g_0} \lambda^\alpha \lambda^{\beta_0} \lambda^{\beta_{j_3}} \lambda^{\beta_{j_4}} \dots \lambda^{\beta_{j_n}} G_0 = \lambda^\alpha.$$



Similarly, for  $\alpha \in \{0, 1\}^k$  and  $\text{wt}(\alpha)$  is odd,

$$\Gamma_\lambda^{\alpha 1} = \frac{1}{g_1} \underline{G}^{\alpha \beta_1 \beta_2 \beta_3 \dots \beta_n} = \frac{1}{g_1} \lambda^\alpha \lambda^{\beta_1} \lambda^{\beta_2} \lambda^{\beta_3} \dots \lambda^{\beta_n} G_1 = \lambda^\alpha.$$

This completes the proof. □

### 4 Collapse Theorem

By (5) and Theorem 5, we have

**Theorem 6.** *If a generator is realizable on a valid basis  $T$ , then it is also realizable on its embedded size 1 basis  $\widehat{T}$ .*

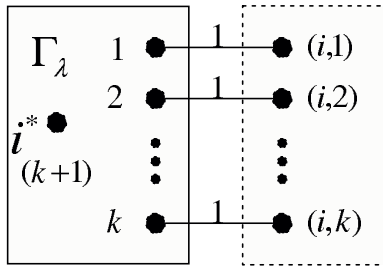
Now we prove the collapse result on the recognizer side.

**Theorem 7.** *If a recognizer  $R$  is realizable on a valid basis  $T$ , then it is also realizable on its embedded size 1 basis  $\widehat{T}$ .*

**Proof:** Since  $T$  is a valid basis, from Section 3, we have its embedded size 1 basis  $\widehat{T}$ , and the tensor  $(\lambda^\alpha)$ . By the proof of Theorem 4 we have an even matchgate  $\Gamma_\lambda$  whose condensed signature is  $\lambda^\alpha$ .

Let  $\Gamma'$  be a matchgate realizing  $\underline{R}$ ,  $R = \underline{R}T^{\otimes n}$ .  $\Gamma'$  has  $kn$  external nodes.

For every block of  $k$  nodes in  $\Gamma'$ , we use the matchgate  $\Gamma_\lambda$  from Section 3 to extend  $\Gamma'$  to get a new matchgate  $\widehat{\Gamma}'$  of arity  $n$  (see Figure 4).



**Fig. 4.** Extend the  $i$ -th block of recognizer  $\Gamma'$  by a copy of  $\Gamma_\lambda$ . We rename the  $(k+1)$ -th node of this copy of  $\Gamma_\lambda$  as  $i^*$ , which is the  $i$ -th external node of the new recognizer  $\widehat{\Gamma}'$ .

The idea is that, for each block of  $k$  external nodes in  $\Gamma'$ , we take one copy of  $\Gamma_\lambda$  and fold it around so that in a planar fashion its first  $k$  external nodes are connected to the  $k$  external nodes in  $\Gamma'$  in this block. The  $(k+1)$ -st external node of this copy of  $\Gamma_\lambda$  becomes a new external node of  $\widehat{\Gamma}'$ . Altogether  $\widehat{\Gamma}'$  has  $n$  external nodes  $1^*, 2^*, \dots, n^*$ .

Since  $\Gamma_\lambda$  is an even matchgate, when the node  $i^*$  is either left in (set to 0) or taken out (set to 1), the only possible non-zero patterns within the  $i$ -th copy of  $\Gamma_\lambda$  are all  $\alpha_i \in \{0, 1\}^k$  with the same parity.

It follows that the following exponential sum holds, for all  $i_1, i_2, \dots, i_n = 0, 1$ :

$$\widehat{R}_{i_1 i_2 \dots i_n} = \sum_{\oplus \alpha_r = i_r} \underline{R}_{\alpha_1 \alpha_2 \dots \alpha_n} \lambda^{\alpha_1} \lambda^{\alpha_2} \dots \lambda^{\alpha_n}.$$

where  $\widehat{R}$  is the standard signature of  $\widehat{\Gamma}'$ , and  $\underline{R}$  is the standard signature of  $\Gamma'$ .

We want to prove that  $\widehat{R}$  in the basis  $\widehat{T} = (\widehat{t}_i) = \left[ \begin{pmatrix} \widehat{n}^0 \\ \widehat{n}^1 \end{pmatrix}, \begin{pmatrix} \widehat{p}^0 \\ \widehat{p}^1 \end{pmatrix} \right]$  and  $\underline{R}$  in the basis  $T = (t_i^\alpha)$  give the same recognizer  $R$ .

Recall that  $t_i^\alpha = \lambda^\alpha \widehat{t}_i^{\oplus \alpha}$ . Now from (2) we have

$$\begin{aligned} R_{l_1 l_2 \dots l_n} &= \sum_{\alpha_r \in \{0,1\}^k} \underline{R}_{\alpha_1 \alpha_2 \dots \alpha_n} t_{l_1}^{\alpha_1} t_{l_2}^{\alpha_2} \dots t_{l_n}^{\alpha_n} \\ &= \sum_{i_r \in \{0,1\}} \sum_{\oplus \alpha_r = i_r} \underline{R}_{\alpha_1 \alpha_2 \dots \alpha_n} t_{l_1}^{\alpha_1} t_{l_2}^{\alpha_2} \dots t_{l_n}^{\alpha_n} \\ &= \sum_{i_r \in \{0,1\}} \sum_{\oplus \alpha_r = i_r} \underline{R}_{\alpha_1 \alpha_2 \dots \alpha_n} \lambda^{\alpha_1} \widehat{t}_{l_1}^{\oplus \alpha_1} \lambda^{\alpha_2} \widehat{t}_{l_2}^{\oplus \alpha_2} \dots \lambda^{\alpha_n} \widehat{t}_{l_n}^{\oplus \alpha_n} \\ &= \sum_{i_r \in \{0,1\}} \widehat{t}_{l_1}^{i_1} \widehat{t}_{l_2}^{i_2} \dots \widehat{t}_{l_n}^{i_n} \sum_{\oplus \alpha_r = i_r} \underline{R}_{\alpha_1 \alpha_2 \dots \alpha_n} \lambda^{\alpha_1} \lambda^{\alpha_2} \dots \lambda^{\alpha_n} \\ &= \sum_{i_r \in \{0,1\}} \widehat{t}_{l_1}^{i_1} \widehat{t}_{l_2}^{i_2} \dots \widehat{t}_{l_n}^{i_n} \widehat{R}_{i_1 i_2 \dots i_n}. \end{aligned}$$

The last equation shows that  $R$  is also the signature of  $\widehat{\Gamma}'$  under basis  $\widehat{T}$ . This completes the proof. □

From Theorems 6 and 7, we can prove the following main theorem. See 7.

**Theorem 8.** (*Bases Collapse Theorem*) Any holographic algorithm on a basis of any size which employs at least one non-degenerate generator can be efficiently transformed to an holographic algorithm in a basis of size 1. More precisely, if generators  $G_1, G_2, \dots, G_s$  and recognizers  $R_1, R_2, \dots, R_t$  are simultaneously realizable on a basis  $T$  of any size, and not all generators are degenerate, then all the generators and recognizers are simultaneously realizable on a basis  $\widehat{T}$  of size 1, which is the embedded basis of  $T$ .

We remark that a holographic algorithm which only uses degenerate generators is trivial. From Theorem 8, what can be computed in P-time by holographic algorithms in arbitrary dimensional bases can also be done with bases of size 1. This rules out infinitely many theoretical possibilities. Regarding holographic algorithms over size 1 basis, we have already built a substantial theory 5. Therefore this is an important step towards the understanding of the ultimate capability of holographic algorithms.

## References

1. Cai, J-Y., Choudhary, V.: Some Results on Matchgates and Holographic Algorithms. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051(Part I), pp. 703–714. Springer, Heidelberg (2006) Also available at Electronic Colloquium on Computational Complexity TR06-048, 2006
2. Cai, J-Y., Choudhary, V.: Valiant’s Holant Theorem and Matchgate Tensors (Extended Abstract). In: Cai, J.-Y., Cooper, S.B., Li, A. (eds.) TAMC 2006. LNCS, vol. 3959, pp. 248–261. Springer, Heidelberg (2006) Also available at Electronic Colloquium on Computational Complexity Report TR05-118
3. Cai, J-Y., Choudhary, V., Lu, P.: On the Theory of Matchgate Computations. To appear in CCC 2007
4. Cai, J-Y., Lu, P.: On Symmetric Signatures in Holographic Algorithms. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 429–440. Springer, Heidelberg (2007)
5. Cai, J-Y., Lu, P.: Holographic Algorithms: From Art to Science. To appear in STOC 2007. Also available at Electronic Colloquium on Computational Complexity Report TR06-145
6. Cai, J-Y., Lu, P.: Bases Collapse in Holographic Algorithms. To appear in CCC 2007. Also available at Electronic Colloquium on Computational Complexity Report TR07-003
7. Cai, J-Y., Lu, P.: Holographic Algorithms: The Power of Dimensionality Resolved. Available at Electronic Colloquium on Computational Complexity Report TR07-020
8. Dodson, C.T.J., Poston, T.: Tensor Geometry. Graduate Texts in Mathematics, 2nd edn., vol. 130. Springer, Heidelberg (1991)
9. Lichtenstein, D.: Planar formulae and their uses. *SIAM J. Comput.* 11(2), 329–343 (2000)
10. Jerrum, M.: Two-dimensional monomer-dimer systems are computationally intractable. *J. Stat. Phys.* 48, 121–134 (1987) erratum. 59, 1087-1088 (1990)
11. Kasteleyn, P.W.: The statistics of dimers on a lattice. *Physica* 27, 1209–1225 (1961)
12. Kasteleyn, P.W.: Graph Theory and Crystal Physics. In: Harary, F. (ed.) *Graph Theory and Theoretical Physics*, pp. 43–110. Academic Press, London (1967)
13. Knill, E.: Fermionic Linear Optics and Matchgates. At <http://arxiv.org/abs/quant-ph/0108033>
14. Murota, K.: *Matrices and Matroids for Systems Analysis*. Springer, Heidelberg (2000)
15. Temperley, H.N.V., Fisher, M.E.: Dimer problem in statistical mechanics – an exact result. *Philosophical Magazine* 6, 1061–1063 (1961)
16. Valiant, L.G.: Quantum circuits that can be simulated classically in polynomial time. *SIAM Journal of Computing* 31(4), 1229–1254 (2002)
17. Valiant, L.G.: Expressiveness of Matchgates. *Theoretical Computer Science* 281(1), 457–471 (2002)
18. Valiant, L.G.: Holographic Algorithms (Extended Abstract). In: Proc. 45th IEEE Symposium on Foundations of Computer Science, pp. 306–315 (2004) A more detailed version appeared in Electronic Colloquium on Computational Complexity Report TR05-099
19. Valiant, L.G.: Holographic circuits. In: Proc. 32nd International Colloquium on Automata, Languages and Programming, pp. 1–15 (2005)
20. Valiant, L.G.: Completeness for parity problems. In: Proc. 11th International Computing and Combinatorics Conference, pp. 1–8 (2005)
21. Valiant, L.G.: Accidental Algorithms. In: Proc. 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 509–517 (2006)

# Reconciling Data Compression and Kolmogorov Complexity

Laurent Bienvenu<sup>1</sup> and Wolfgang Merkle<sup>2</sup>

<sup>1</sup> Laboratoire d'Informatique Fondamentale, Université de Provence, Marseille, France  
laurent.bienvenu@lif.univ-mrs.fr

<sup>2</sup> Institut für Informatik, Ruprecht-Karls-Universität Heidelberg, Germany  
merkle@math.uni-heidelberg.de

**Abstract.** While data compression and Kolmogorov complexity are both about effective coding of words, the two settings differ in the following respect. A compression algorithm or compressor, for short, has to map a word to a unique code for this word in one shot, whereas with the standard notions of Kolmogorov complexity a word has many different codes and the minimum code for a given word cannot be found effectively. This gap is bridged by introducing decidable Turing machines and a corresponding notion of Kolmogorov complexity, where compressors and suitably normalized decidable machines are essentially the same concept.

Kolmogorov complexity defined via decidable machines yields characterizations in terms of the initial segment complexity of sequences of the concepts of Martin-Löf randomness, Schnorr randomness, Kurtz randomness, and computable dimension. These results can also be reformulated in terms of time-bounded Kolmogorov complexity. Other applications of decidable machines are presented, such as a simplified proof of the Miller-Yu theorem (characterizing Martin-Löf randomness by the plain complexity of the initial segments) and a new characterization of computably traceable sequences via a natural lowness notion for decidable machines.

## 1 Introduction

The Kolmogorov complexity of a word  $w$  with respect to a Turing machine  $M$  is defined to be the length of the shortest input on which  $M$  halts and outputs  $w$ . This shortest input is often thought as the “best compression” of  $w$  relatively to  $M$ . However, as pointed out by Shen [14]: “[...] in the framework of Kolmogorov complexity we have no compression algorithm and deal only with the decompression algorithm.” In this paper, we address Shen’s remark and have a look at effective randomness from an actual compression viewpoint.

First, in Section 2, we introduce a formal notion of compressor. Among the many possible formalizations of the notion of compressor, we study one that is natural and captures the main properties of “real world” compressors. We then argue that compression obtained by compressors in our sense is essentially the same as dealing with Kolmogorov complexity defined via *decidable machines*, i.e., by Turing machines that have computable domains.

In the following sections, the theory of decidable machines is developed in a framework of algorithmic randomness. In Section 3, we review the celebrated characterization of Martin-Löf randomness in terms of prefix-free Kolmogorov complexity due to Schnorr and the more recent characterizations of Schnorr and Kurtz randomness in terms of bounded machines due to Downey and Griffiths [2]. We give identical or very similar characterizations of all three notions of randomness in terms of decidable machines; to the best of our knowledge, this is the first time that all three notions are characterized using a single type of Turing machine. Similarly, we argue that the characterization of computable Hausdorff dimension in terms of computable machines due to Downey et al. [1] extends to decidable machines. In Section 4, all the mentioned characterizations are transferred to standard time-bounded Kolmogorov complexity by arguing that the latter is closely related to Kolmogorov complexity defined via decidable machines.

In Section 5, we use the characterization of Martin-Löf randomness in terms of decidable machines in order to give a simplified proof of the recent Miller-Yu theorem, which characterizes Martin-Löf randomness in terms of plain Kolmogorov complexity of initial segments.

Finally, in Section 6, we consider lowness notions. A sequence  $A$  is called low and order-low for prefix-free decidable machines in case Kolmogorov complexity with respect to such machines, when relativized to oracle  $A$ , changes by less than a constant and by less than any computable order, respectively. We obtain that any sequence is order-low for decidable machines if the sequence is computably traceable, which then implies some known facts on lowness for Schnorr and Kurtz randomness. Furthermore, exactly the computable sequences are low for prefix-free decidable machines. In what follows several proofs will be omitted due to lack of space.

We conclude the introduction by reviewing some standard concepts and notation that will be used in the sequel. A **WORD** is a finite binary sequence, the empty word, the unique word of length 0, is denoted by  $\lambda$ ; the set of all words is denoted by  $2^*$ . Unless explicitly stated differently, a **SEQUENCE** is an infinite binary sequence, and we write  $2^\omega$  for the set of all sequences (sometimes also referred to as Cantor space). We denote by  $\sqsubseteq$  the prefix relation on  $2^* \cup 2^\omega$ . For every word  $u$  and for every set of words  $A$ , let

$$[u] = \{R \in 2^\omega : u \sqsubseteq R\} \quad \text{and} \quad [A] = \bigcup_{u \in A} [u].$$

Recall that Lebesgue measure on Cantor space is the unique probability measure  $\mu$  on Cantor space such that for all words  $u$  holds  $\mu([u]) = 2^{-|u|}$ .

Following Li and Vitanyi [9], we denote by  $C(u)$  and  $K(u)$  the plain and the prefix Kolmogorov complexity of a word  $u$ . For every Turing machine  $M$  which computes a partial function from words to words and for any word  $u$ , we denote by  $C_M(u)$  the natural number  $\inf\{|p| : M(p) = u\}$ . If  $M$  is assumed to have prefix-free domain, we write  $K_M$  in place of  $C_M$ .

## 2 Compressors and Decidable Machines

The intuitive understanding of a compressor is a procedure that maps a word to a code for that word, where the mapping is one-to-one and hence in principle invertible. For compressors that are to be applied in practice, in addition one will surely require that coding and decoding are efficient and that redundant sources will be mapped to reasonably short codes; however, these latter requirements will not be considered here. We consider a most general notion of compressor where one simply requires that the compressor yields an effective bijection between a computable set of words and a computable set of codes.

**Definition 1.** A COMPRESSOR is a partial computable function  $\Gamma: 2^* \rightarrow 2^*$  such that  $\Gamma$  is one-to-one and the domain and range of  $\Gamma$  are computable. A compressor is PREFIX-FREE if its range is prefix-free.

By Definition 1, a compressor may be undefined on some strings, but such strings can be recognized effectively. Furthermore, the definition ensures that decompression can be performed effectively, i.e., given a string  $u = \Gamma(v)$  in the range of  $\Gamma$ , the unique preimage  $v$  of  $u$  can be found effectively. Here again, some strings can be invalid for the decompression process, i.e., they do not represent a compressed string, but these strings can also be recognized effectively. Most actual compressors (e.g., gzip) are indeed compressors in our sense.

Compressors as just defined and Kolmogorov complexity both are about effective coding of words. The main difference is that a compressor has to produce a code for a word in one shot and in fact every word has at most a single code. For the standard notions of Kolmogorov complexity, on the other hand, a word has several codes. While decoding is effective and the set of pairs of codes and coded words is computably enumerable, in general there is no effective way to go from a word to its shortest code. So if we want to have a notion of Kolmogorov complexity that corresponds to the type of coding done by compressors, we have to define Kolmogorov complexity with respect to a restricted class of Turing machines.

**Definition 2.** A Turing machine that computes a partial function on the set of words is called DECIDABLE if its domain is decidable.

Decidable Turing machines can be normalized in the sense that superfluous codes are removed and possessing a code becomes a computable property.

**Definition 3.** A decidable Turing machine  $M$  is called NORMALIZED if the range of  $M$  is decidable and for all words  $w$  there is at most one word  $p$  where  $M(p)=w$ .

As usual, a Turing machine is called PREFIX-FREE if its domain is prefix-free.

**Proposition 4.** Given a decidable Turing machine  $M$  one can effectively find a decidable Turing machine  $M'$  that is normalized such that for all words  $w$  holds  $K_{M'}(w) \leq K_M(w) + 1$ . In addition, if  $M$  is prefix-free, then  $M'$  can be chosen to be prefix-free, too.

*Proof.* For a given decidable Turing machine  $M$ , define  $M'$  as follows. For any word  $w$ , in case there is some word  $p$  where  $M(p) = w$  and  $|p| \leq 2|w|$ , let  $p_w$  be the least such word and let  $M'(0p_w) = w$ . If there is no such word  $p$ , let  $M'(1^{|w|}0w) = w$ .  $\square$

Proposition 5 shows that compressors and normalized decidable Turing machines are essentially the same. Recall that the inverse of a partial function  $f$  from words to words that is one-to-one is the partial function  $f^{-1}: v \mapsto \min\{u: f(u) = v\}$ .

**Proposition 5.** *For any compressor  $\Gamma$ , the inverse mapping  $\Gamma^{-1}$  is just the partial function computed by some normalized decidable Turing machine. For any normalized decidable Turing machine  $M$  the inverse mapping  $M^{-1}$  is a compressor. The same correspondence is given for the subclasses of prefix-free compressors and prefix-free decidable machines.*

*Proof.* It suffices to observe that both compressors and normalized decidable Turing machines are by definition just partial computable bijections between a decidable domain and a decidable range.  $\square$

In the sequel, we will derive several results on decidable Turing machines, however, by the close correspondence with compressors, all these results could be reformulated in terms of compressors. For further use, recall the notion of computable Turing machine, as introduced by Downey and Griffiths.

**Definition 6 (Downey and Griffiths).** *A Turing machine that computes a partial function on the set of words is called COMPUTABLE if its domain  $D$  is prefix-free and  $[D]$  has computable Lebesgue measure.*

Observe that any computable Turing machine is in particular decidable; the converse is false even when attention is restricted to prefix-free Turing machines. Downey and Griffiths [2] introduced the notion of computable machine in order to give a machine characterization of Schnorr randomness and Kurtz randomness. In Section 3, we will give alternative characterizations of these randomness notions in terms of decidable machines.

### 3 Characterizing Randomness Notions by Decidable Machines

We first review the notions of a Martin-Löf random, a Schnorr random, and a Kurtz random sequence, and then we characterize these notions in terms of the complexity of initial segments with respect to decidable machines.

**Definition 7.** *A MARTIN-LÖF TEST is a uniformly recursively enumerable sequence  $V_0, V_1, \dots$  of sets of words where  $[V_n]$  has Lebesgue measure at most  $2^{-n}$ . A SCHNORR TEST is a Martin-Löf test where in addition the Lebesgue measure of  $[V_n]$  is exactly  $2^{-n}$ . A KURTZ TEST is a Martin-Löf test where in addition the sets  $[V_n]$  are finite and a canonical index for  $V_n$  can be computed from  $n$ .*

A sequence  $V_0, V_1, \dots$  of sets of words *COVERS* a sequence  $X$  if  $X$  is contained in the intersection of the open sets  $[V_n]$  (i.e., if  $X$  has a prefix in every set  $V_i$ ). A sequence  $R$  is *MARTIN-LÖF RANDOM* if it cannot be covered by a Martin-Löf test. A sequence  $R$  is *SCHNORR RANDOM* if it cannot be covered by a Schnorr test. A sequence  $R$  is *KURTZ RANDOM* if it cannot be covered by a Kurtz test.

In the above definition of a Schnorr test, the measure condition on the uniformly recursively enumerable family  $V_0, V_1, \dots$  actually implies that this family is uniformly computable. This is no longer true in the case of Martin-Löf tests: some are not uniformly computable. However, it is well-known that every Martin-Löf test can be turned into an equivalent Martin-Löf test that is uniformly computable.

**Lemma 8.** *For every Martin-Löf test  $V_0, V_1, \dots$  there exists a Martin-Löf test  $U_0, U_1, \dots$  such that  $[U_n] = [V_n]$  for all  $n$  and where the set  $\{(u, n) : u \in V_n\}$  is decidable. (We will call a Martin-Löf test with the latter property a *DECIDABLE Martin-Löf test*.)*

One of the most celebrated results in algorithmic randomness is Schnorr’s characterization of Martin-Löf randomness in terms of prefix-free Kolmogorov complexity. Propositions 10 and 11 assert that very similar characterizations are true in a setting of prefix-free decidable machines.

**Theorem 9 (Schnorr).** *A sequence  $R$  is Martin-Löf random if and only if  $K(R[0..n]) \geq n - O(1)$ .*

**Proposition 10.** *A sequence  $R$  is Martin-Löf random if and only if for all prefix-free decidable machines  $M$ ,  $K_M(R[0..n]) \geq n - O(1)$ .*

**Proposition 11.** *There exists a prefix-free decidable machine  $M$  such that any sequence  $R$  is Martin-Löf random if and only if  $K_M(R[0..n]) \geq n - O(1)$ .*

The implications from left to right in Propositions 10 and 11 are immediate from Schnorr’s theorem because for any prefix-free machine  $M$  there is a constant  $d$  such that for all words  $u$  holds  $K(u) \leq K_M(u) + d$ . The proof of the other direction is similar to the proof of the corresponding implication in Schnorr’s theorem, i.e., one applies the Kraft-Chaitin Theorem in order to go from a Martin-Löf test that covers a set  $X$  to a prefix-free machine that has short codes for the prefixes of  $X$ ; in order to actually obtain prefix-free decidable machines, it suffices to start with a decidable Martin-Löf test according to Lemma 8, where in the case of Proposition 11 this test is chosen to be universal.

Downey and Griffiths 2 characterized Schnorr randomness by the complexity of initial segments with respect to computable machines; Proposition 13 states a related characterization in terms of decidable Turing machines. In connection with the latter, recall that a function  $h: \mathbb{N} \rightarrow \mathbb{N}$  is called an *ORDER* if  $h$  is nondecreasing and unbounded.

**Proposition 12 (Downey and Griffiths).** *A sequence  $R$  is Schnorr random iff for every computable Turing machine  $M$  holds  $K_M(R[0..n]) \geq n + O(1)$ .*



**Proposition 13.** *A sequence  $R$  is Schnorr random iff for every computable order  $h$  and for every prefix-free decidable Turing machine  $M$ ,*

$$K_M(R[0..n]) \geq n - h(n) + O(1) .$$

Downey and Griffiths [2] also used computable machines to give a machine characterization of Kurtz randomness. Proposition [15] asserts a completely similar characterization in terms of decidable machines, where interestingly the prefix-free property makes no difference. Furthermore, the last condition in Proposition [15] gives a characterization of Kurtz randomness similar to the characterization of Schnorr randomness in Proposition [13].

**Proposition 14 (Downey and Griffiths)**

*The following assertions are equivalent.*

- (i)  *$R$  is not Kurtz random.*
- (ii) *There exists a computable machine  $M$  and a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $K_M(R[0..f(d)]) \leq f(d) - d$  holds for all  $d$ .*

**Proposition 15.** *The following assertions are equivalent.*

- (i)  *$R$  is not Kurtz random.*
- (ii) *There exists a prefix-free decidable machine  $M$  and a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $K_M(R[0..f(d)]) \leq f(d) - d$  holds for all  $d$ .*
- (iii) *There exists a decidable machine  $M$  and a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $C_M(R[0..f(d)]) \leq f(d) - d$  holds for all  $d$ .*
- (iv) *There exists a decidable machine  $M$  and a computable order  $h$  such that  $C_M(R[0..n]) \leq n - h(n)$  holds for all  $n$ .*

A characterization of computable dimension in terms of computable machines has been obtained by Downey et al. [11], and we conclude this section by a similar characterization in terms of decidable machines. In the context of time-bounded complexity, an equivalent formulation of the latter characterization has been previously demonstrated by Hitchcock [5], see Proposition [20].

**Proposition 16.** *For every sequence  $R$  holds*

$$\dim_{\text{comp}}(R) = \inf_M \liminf_{n \rightarrow +\infty} \frac{C_M(R[0..n])}{n} ,$$

*where the infimum is over all decidable Turing machines  $M$ .*

## 4 Time-Bounded Kolmogorov Complexity

Since a decidable Turing machine is required to have a computable domain, it is not hard to show that a decidable Turing machine is the same as a Turing machine that obeys a time bound  $t$  but is not required to be defined on all inputs, i.e., on any input  $p$  the machine runs for at most  $t(|p|)$  steps and then either

produces an output or alternatively may decide not to terminate. In contrast to this, time-bounded Kolmogorov complexity is usually defined in terms of time-bounded machines where the time-bound is required with respect to the length of the output, that is the machine may again be undefined on certain inputs, but whenever the machine outputs a word  $w$  then the corresponding computation runs for at most  $t(|w|)$  many steps.

**Definition 17.** Fix an additively optimal machine  $U$ . For any computable function  $t : \mathbb{N} \rightarrow \mathbb{N}$  and any word  $w$ , let

$$C^t(w) = \min\{|p| : U(p) \text{ outputs } w \text{ after at most } t(|w|) \text{ steps of computation}\}.$$

$K^t(w)$  is defined similarly, taking  $U$  optimal among the prefix-free machines.

**Lemma 18.** For every decidable machine  $M$ , there exists a computable time bound  $t$  such that  $C^t \leq C_M + O(1)$ . For every prefix-free decidable machine  $M$ , there exists a computable time bound  $t$  such that  $K^t \leq K_M + O(1)$ .

The converse of Lemma 18 is not true, but the following weaker statement will be sufficient for our purposes.

**Lemma 19.** For every computable time bound  $t$  and every computable order  $h$ , there exists a decidable machine  $M$  such that for every word  $w$  and all  $k \in \mathbb{N}$ ,

$$C^t(w) \leq |w| - k \implies C_M(w) \leq |w| - k + h(k).$$

A similar statement holds for  $K^t$  and prefix-free decidable machines.

By Lemma 18 and Lemma 19, all our previous results can be interpreted in terms of time-bounded Kolmogorov complexity.

**Proposition 20.** (a) A sequence  $R$  is Martin-Löf random iff for every computable time bound  $t$ ,  $K^t(R[0..n]) \geq n + O(1)$ .

(b) There exists a computable time bound  $t_0$  such that every sequence  $R$  is Martin-Löf random iff  $K^{t_0}(R[0..n]) \geq n + O(1)$ .

(c) A sequence  $R$  is Schnorr random iff for every computable time bound  $t$  and every computable order  $g$ ,  $K^t(R[0..n]) \geq n - g(n) + O(1)$ .

(d) A sequence  $R$  is Kurtz random iff for every computable time bound  $t$  and every computable order  $g$ ,  $K^t(R[0..n]) \geq n - g(n)$  for infinitely many  $n$  (and this equivalence remains true with  $C^t$  in place of  $K^t$ ).

(e) For every sequence  $R$ ,  $\dim_S(R) = \inf \liminf \frac{K^t(R[0..n])}{n}$ , the infimum being taken over all computable time bounds  $t$  (and this equation remains true with  $C^t$  in place of  $K^t$ ).

Assertion (e) was proved earlier by Hitchcock 5. Assertion (c) is an improvement of a result of Lathrop and Lutz 8, who demonstrated that the right-hand side condition is necessary for  $R$  to be computably random.

## 5 The Miller-Yu Theorem

After Schnorr [3] characterized Martin-Löf randomness in terms of the prefix Kolmogorov complexity of initial segments, the question whether there is a similar characterization in terms of plain complexity remained open for more than three decades until recently Miller and Yu [11] gave a positive answer. A simplified proof of their result is obtained by using the characterization of Martin-Löf randomness via prefix-free decidable machines from Proposition 11.

**Proposition 21 (Miller and Yu).** *There is a computable function  $G: \mathbb{N} \rightarrow \mathbb{N}$  such that the sum  $\sum_{n \in \mathbb{N}} 2^{-G(n)}$  converges and such that for any sequence  $R$  the following assertions are equivalent.*

- (i)  $R$  is Martin-Löf random.
- (ii) For every computable function  $g: \mathbb{N} \rightarrow \mathbb{N}$  such that  $\sum_{n \in \mathbb{N}} 2^{-g(n)}$  converges it holds that  $C(R[0..n-1]) \geq n - g(n) - O(1)$ .
- (iii) It holds that  $C(R[0..n-1]) \geq n - G(n) - O(1)$ .

*Proof.* For completeness, we review the standard proof of the implication (i)→(ii). If (ii) is false, then there is a computable function  $g$  where  $\sum_{n \in \mathbb{N}} 2^{-g(n)}$  converges and such that for arbitrarily large  $d$  there is some  $n$  where

$$C(R[0..n-1]) \leq n - g(n) - d. \tag{1}$$

But for any such  $d$  and  $n$ , inequality (1) remains true with  $g(n)$  replaced by  $K(n)$  because  $K(n) \leq g(n) + O(1)$  holds by the Kraft-Chaitin Theorem 3 and assumption on  $g$ . Hence for any such  $d$  and  $n$ , the prefix  $w$  of  $R$  of length  $n$  has a prefix-free code of length at most  $n - d/2 + O(1)$ , which consists of a prefix-free code for  $n$  of length  $K(n)$ , followed by a prefix-free code for  $n - K(n) - C(w)$  plus a plain code for  $w$  of length  $C(w)$ . Consequently,  $R$  is not Martin-Löf random.

We now construct a computable function  $G$  with the required convergence property, where the implication (ii)→(iii) is then immediate, and we conclude by giving an alternative proof of (iii)→(i). Let  $M$  be the prefix-free decidable machine of Proposition 11. For all  $n, c \in \mathbb{N}$ , let  $A_n^c = \{u: |u| = n \text{ and } K_M(u) \leq |u| - c\}$  and  $a_n^c = \text{Card}(A_n^c)$ . Furthermore, let  $b_n^c = 2^c a_n^c$  and  $b_n = \sum_{c \in \mathbb{N}} b_n^c$ ; observe that the sums of the latter type are actually finite because  $A_n^c$  is empty for  $c > n$ . This way we have

$$\sum_{n \in \mathbb{N}} b_n \frac{1}{2^n} = \sum_{n, c \in \mathbb{N}} b_n^c \frac{1}{2^n} = \sum_{n, c \in \mathbb{N}} a_n^c \frac{1}{2^{n-c}} \leq 1, \tag{2}$$

where the equalities hold by definition and the inequality holds because  $M$  is prefix-free. Now, if we let  $G(n) = n - \log(b_1 + \dots + b_n)$ , then  $G$  is computable and by definition of  $G$  and elementary rearrangements of terms one obtains

$$\sum_{n \in \mathbb{N}} 2^{-G(n)} \leq \sum_{n \in \mathbb{N}} \frac{b_1 + \dots + b_n}{2^n} \leq 2 \sum_{n \in \mathbb{N}} \frac{b_n}{2^n} \leq 2.$$

Next consider any word  $w$  in  $A_n^c$ . Since the  $A_n^c$ 's are uniformly computable, the word  $w$  can be obtained effectively from a description that contains  $c$  together with the index of  $w$  in the enumeration of the union of the sets  $A_0^c, A_1^c, \dots$  where the elements of  $A_n^c$  are enumerated before those of  $A_{n+1}^c$ . Hence it holds that

$$\begin{aligned} C(w) &\leq 2 \log c + \log(a_1^c + a_2^c \dots + a_n^c) + O(1) \\ &\leq 2 \log c + \log(2^{-c}b_1 + 2^{-c}b_2 + \dots + 2^{-c}b_n) + O(1) \\ &\leq n - G(n) - c + 2 \log c + O(1) \leq n - G(n) - c/2 + O(1), \end{aligned} \tag{3}$$

where the second and third inequality hold by definition of the  $b_n$ 's and of  $G$ , respectively. Now if the sequence  $R$  is not Martin-Löf random, then by Proposition 11 and definition of the  $A_n^c$ 's, there are arbitrarily large  $c$  such that for some  $n$  the prefix  $w$  of  $R$  of length  $n$  is in  $A_n^c$  and thus  $w$  and  $c$  satisfy the chain of inequalities (3), hence (iii) is false. □

## 6 Lowness and Order-Lowness

In the area of algorithmic randomness, various lowness notions have been studied and have been shown to interact interestingly with each other and with other notions [3,12]. In general, a sequence  $A$  is called low for a certain concept, if the concept does not change (or at least does not change significantly) when the concept is relativized to the sequence  $A$ . For example, a sequence  $A$  is low for  $K$  if the the standard prefix-free Kolmogorov complexity  $K$  and it relativized version with oracle  $A$  differ at most by some additive constant. In connection with complexity notions that are not defined via a universal machine, such as Kolmogorov complexity defined in terms of prefix-free decidable machines, lowness usually means that for any machine of the given type with oracle there is another machine without oracle such that complexity with respect to the latter is not significantly larger than complexity with respect to the former. Note in this connection that a prefix-free decidable machine with oracle  $A$  is an oracle Turing machine with oracle  $A$  that on oracle  $A$  has prefix-free domain that can be computed with oracle  $A$ ; we write  $K_M^A$  for the corresponding relativized notion of Kolmogorov complexity.

**Definition 22.** *A sequence  $A$  is LOW FOR PREFIX-FREE DECIDABLE MACHINES if for every prefix-free decidable machine  $M$  with oracle  $A$ , there exists a prefix-free decidable machine  $M'$  such that for all words  $w$ ,*

$$K_{M'}(w) \leq K_M^A(w) + O(1).$$

*A sequence  $A$  is ORDER-LOW FOR PREFIX-FREE DECIDABLE MACHINES if for every prefix-free decidable machine  $M$  with oracle  $A$  and any computable order  $h$ , there exists a prefix-free decidable machine  $M'$  such that for all words  $w$ ,*

$$K_{M'}(w) \leq K_M^A(w) + h(K_M^A(w)) + O(1). \tag{4}$$

The notion of order-low is similar to a lowness notion for standard prefix-free Kolmogorov complexity that has been introduced and has been shown to be equivalent to strong jump-traceability by Figueira, Nies, and Stephan [4]. Somewhat similar to their equivalence result, we obtain a characterization of computable traceability in terms of order-lowness for prefix-free decidable machines. Recall the concept of computable traceability.

**Definition 23.** *A sequence  $A$  is computably traceable if there is a computable order  $h$  such that for any function  $f$  that is computable with oracle  $A$  there is a computable sequence of canonical indices for finite sets  $F_0, F_1, \dots$  where for all  $i$ ,*

$$(i) \quad f(i) \in F_i, \quad \text{and} \quad (ii) \quad |F_i| \leq h(i).$$

**Proposition 24.** *A sequence  $A$  is computably traceable if and only if  $A$  is order-low for prefix-free decidable machines.*

Terwijn and Zambella [16] observed that the notion of computable traceability is robust in the following sense. The notion remains the same if in its definition it is required that for any computable order  $h$  and any function  $f$  computable in  $A$  there is a trace  $F_0, F_1, \dots$  with  $f(i) \in F_i$  and  $|F_i| \leq h(i)$ ; that is, if according to Definition 23 a sequence  $A$  is computably traceable with respect to some computable order  $h$ , then the sequence is computably traceable with respect to any computable order  $h$ . Indeed, the definition holds then even with respect to any order that is computable in  $A$ , because by Remark 25, for any computably traceable sequence  $A$  and any order  $g$  computable in  $A$  there is a computable order  $h$  where  $h(n) \leq g(n)$  for all  $n$ .

Recall that by definition sequence  $A$  is hyperimmune-free, if for any function  $f$  that is computable with oracle  $A$  there is a computable function  $g$  such that  $f(n) \leq g(n)$  for all  $n$ . Furthermore, observe that every computably traceable sequence  $A$  is hyperimmune-free, where it suffices to let  $g(n)$  be the maximum value in  $F_i$  where  $F_0, F_1, \dots$  is a computable trace for a given function  $f$  that is computable in  $A$ .

**Remark 25.** *For any hyperimmune-free sequence  $A$ , thus in particular for any computably traceable sequence  $A$ , and for any order  $h$  that is computable in  $A$  there is a computable order  $g$  such that for all  $n$  holds  $g(n) \leq h(n)$  (in fact, this property characterizes the hyperimmune-free sequences).*

*For a proof,  $h$  being given, consider the  $A$ -computable function  $f$  defined by:  $f(k) = \max\{n : h(n) \leq k\}$ . If  $A$  is hyperimmune-free, there is a computable function  $f'$  such that for all  $n$ ,  $f(n) \leq f'(n)$ , and we can assume that  $f'$  is increasing. Set  $g(k) = \max\{n : f'(n) \leq k\}$  for all  $k$ . Since  $f'$  is computable and increasing,  $g$  is a computable order. Moreover, for all  $n$ ,  $g(n) \leq h(n)$ . Indeed, suppose that for some  $n$ , one has  $g(n) > h(n)$ . Then:*

$$n \geq f'(g(n)) > f'(h(n)) \geq f(h(n)) \geq n$$

*(the inequalities following from the definition of  $g$ , the fact that  $f'$  is increasing, the fact that  $f' \geq f$  and the definition of  $f$  respectively), a contradiction.*

**Remark 26.** *The concept of order-low for prefix-free decidable machines is not changed if one requires in its definition just that (4) is satisfied for some fixed computable order in place of all such orders. For a proof, observe that the equivalence in Proposition 24 extends to the altered concepts by literally the same proof, and conclude using Remark 25.*

**Remark 27.** *The concept of order-low for prefix-free decidable machines is not changed if one requires in the definition of a sequence  $A$  being order-low that (4) is not just satisfied for all computable orders but for all orders that are computable with oracle  $A$ . For a proof it suffices to observe that if a sequence  $A$  is order-low, then  $A$  is computably traceable by Proposition 24, hence by Remark 25 for any  $A$ -computable order there is a computable order that grows at least as slowly.*

Proposition 24 and the assertions of Remarks 26 and 27 remain true when considering dedicable machines (and corresponding notions of order lowness) in place of prefix-free decidable machines.

Terwijn and Zambella [16] and Kjos-Hanssen, Nies, and Stephan [7] demonstrated that the class of computably traceable sequences coincides with the class of sequences that are low for Schnorr null tests and with the class of sequences that are low for Schnorr randomness, respectively. Furthermore, it is known that the computably traceable sequences form a strict subclass of the class of sequences that are low for Kurtz randomness [15]. We obtain part of these results as a corollary to Propositions 13, 15, and 24.

**Corollary 28.** *Any computably traceable sequence  $A$  is low for Schnorr randomness and low for Kurtz randomness.*

*Proof.* Let  $A$  be a computably traceable sequence, and  $R$  a Schnorr random sequence. We shall prove that  $R$  is  $A$ -Schnorr random. Let  $M$  be a prefix-free decidable machine with oracle  $A$ , and  $h$  be an  $A$ -computable order. Up to normalizing  $M$  as we did in the proof of Proposition 4, let us suppose that  $K_M^A(w) \leq 2|w|$  for all words  $w$ . By Remark 25, there exists a computable order  $g$  such that  $g \leq h/2$ . Set  $g'(n) = g(n/2)$  for all  $n$ , and notice that  $g'$  is a computable order. By Proposition 24, let  $M'$  be a prefix-free decidable machine such that  $K_{M'} \leq K_M^A + g'(K_M^A) + O(1)$ . Since  $R$  is Schnorr random, by Proposition 13,  $K_{M'}(R[0..n]) \geq n - g(n) - O(1)$  for all  $n$ . Hence

$$\begin{aligned} K_M^A(R[0..n]) &\geq n - g(n) - g'(K_M^A(R[0..n])) - O(1) \\ &\geq n - g(n) - g'(2n) - O(1) \\ &\geq n - 2g(n) - O(1) \\ &\geq n - h(n) - O(1) \end{aligned}$$

By Proposition 13 (relativized to  $A$ ),  $R$  is  $A$ -Schnorr random. The proof for Kurtz randomness is similar. □

Finally, the following proposition shows that, in contrast to order-lowness, lowness for decidable machines is a trivial lowness notion.

**Proposition 29.** *A sequence is low for prefix-free decidable machines if and only if the sequence is computable.*

*Proof.* Let  $A$  be a sequence that is low for prefix-free decidable machines. By Proposition 10,  $A$  is low for Martin-Löf randomness, hence  $\Delta_2^0$  (see 12). Moreover, since  $A$  is low for prefix-free decidable machines, it is in particular order-low, which by Proposition 24 implies that  $A$  is computably traceable. As a  $\Delta_2^0$  computably traceable sequence is necessarily computable, we are done.  $\square$

## References

1. Downey, R., Merkle, W., Reimann, J.: Schnorr dimension. *Mathematical Structures in Computer Science* 16, 789–811 (2006)
2. Downey, R., Griffiths, E.: On Schnorr randomness. *Journal of Symbolic Logic* 69(2), 533–554 (2004)
3. Downey, R., Hirschfeldt, D.: Algorithmic Randomness and Complexity. Manuscript (2007)
4. Figueira, S., Nies, A., Stephan, F.: Lowness properties and approximations of the jump. In: WOLLIC 2005. *Electronic Notes in Theoretical Computer Science*, vol. 143, pp. 45–57. Elsevier, Amsterdam (2006)
5. Hitchcock, J.M.: PhD dissertation, Iowa State University (2003)
6. Hitchcock, J.M., Lutz, J.H.: Why computational complexity requires stricter martingales. *Theory of computing systems* 39(2), 277–296 (2006)
7. Kjos-Hanssen, B., Nies, A., Stephan, F.: Lowness for the class of Schnorr random reals. *SIAM Journal on Computing* 35(3), 647–657 (2005)
8. Lathrop, J., Lutz, J.: Recursive computational depth. *Information and Computation* 153, 137–172 (1999)
9. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 2nd edn. Springer, Heidelberg (1997)
10. Martin-Löf, P.: The definition of random sequences. *Information and Control* 9(6), 602–619 (1966)
11. Miller, J.S., Yu, L.: On initial segment complexity and degrees of randomness. *Transactions of the American Mathematical Society* (To appear)
12. Nies, A.: Lowness properties and randomness. *Advances in Mathematics* 197, 274–305 (2005)
13. Schnorr, C.-P.: A unified approach to the definition of random sequences. *Mathematical Systems Theory* 5, 246–258 (1971)
14. Shen, A.: Algorithmic Information Theory and Kolmogorov Complexity. Lecture notes of an introductory course. Uppsala University Technical Report, 2000-034, Available online at <http://www.it.uu.se/publications/reports/2000-034>.
15. Stephan, F., Yu, L.: Lowness for weakly 1-generic and Kurtz-random. In: Cai, J.-Y., Cooper, S.B., Li, A. (eds.) TAMC 2006. LNCS, vol. 3959, pp. 756–764. Springer, Heidelberg (2006)
16. Terwijn, S., Zambella, D.: Computable randomness and lowness. *Journal of Symbolic Logic* 66(3), 1199–1205 (2001)

# Size Competitive Meshing Without Large Angles<sup>\*</sup>

Gary L. Miller, Todd Phillips, and Donald Sheehy

Computer Science Department,  
Carnegie Mellon University, Pittsburgh, PA 15213  
{g1miller,tp517,dsheehy}@cs.cmu.edu

**Abstract.** We present a new meshing algorithm for the plane, Overlay Stitch Meshing (OSM), accepting as input an arbitrary Planar Straight Line Graph and producing a triangulation with all angles smaller than  $170^\circ$ . The output triangulation has competitive size with any optimal size mesh having equally bounded largest angle. The competitive ratio is  $O(\log(L/s))$  where  $L$  and  $s$  are respectively the largest and smallest features in the input. OSM runs in  $O(n \log(L/s) + m)$  time/work where  $n$  is the input size and  $m$  is the output size. The algorithm first uses Sparse Voronoi Refinement to compute a quality overlay mesh of the input points alone. This triangulation is then combined with the input edges to give the final mesh.

## 1 Introduction

The meshing problem is to take as input a domain containing a collection of features and return a triangulation of the domain. In 2D, the features are simply points and non-crossing edges; a planar straight line graph (PSLG). The design of a meshing algorithm involves analysis of four fundamental properties of the algorithm and its outputs. First, the output mesh must be **conforming**, so that all the vertices and edges should appear as a union of simplices in the final mesh. Secondly, all the triangular elements should have some guarantee of **element quality**. Third, the number of output triangles should be asymptotically competitive with any optimal triangulation that is conforming and good quality, so that we have an output size **competitive algorithm**. Finally the algorithm should be **work efficient**. This paper will only be concerned with the 2D meshing problem.

The 2D meshing problem was first posed by Bern, Eppstein, and Gilbert [1] who proposed a quadtree algorithm. Ruppert [2] gave an  $O(n^2)$  time constant size competitive algorithm for the meshing problem using Delaunay refinement. Mitchell and Vavasis [3] extended the quadtree algorithm to 3D and proved that the Bern, Eppstein, and Gilbert algorithm was in fact also a constant size competitive algorithm. In order for these algorithms to be constant factor competitive algorithms, two critical assumptions are made. First, element quality is

---

<sup>\*</sup> This work was supported in part by the National Science Foundation under grants CCR-0122581.



defined by the absence of arbitrarily small angles in any triangle. Secondly, an input PSLG with small angles is not permitted. Our main goal is to develop algorithms and techniques that will allow an arbitrary PSLG as input, relaxing the latter assumption. Note that this also forces us to abandon the former assumption, since any small input angle will require new small output angles to conform [4].

An alternative setting employs a definition of element quality that allows small angles, but bounds all angles strictly away from  $180^\circ$ , prohibiting arbitrarily large angles. Many meshing applications require only this weaker quality guarantee. Babuška and Aziz showed that only large angles affect interpolation error [5, 6], while Boman, Hendrickson, and Vavasis showed that only large angles effect their reduction of a elliptic problem to the solution of a Laplacian problem [7, 8]. An algorithm operating under weaker quality constraints is accordingly able to produce meshes with fewer added vertices (in theory and practice).

In this work, we present and analyze the Overlay Stitch Meshing algorithm (OSM), a new no-large-angle meshing algorithm for conforming to an arbitrary PSLG in 2D. For practical inputs, OSM achieves the best known competitive guarantees on output size relative to the optimal no-large-angle conforming mesh. The algorithm outputs a mesh with no angle larger than  $170^\circ$ . In addition to having good theoretical guarantees, the algorithm is remarkably straightforward and runs in time  $O(n \log(L/s) + m)$ .

## 1.1 Preliminaries

We will use a few standard definitions in our analysis.

**Definition 1.** The **spread**,  $L/s$ , of a PSLG is the ratio of the input diameter ( $L$ ) to the smallest distance between any two disjoint input features ( $s$ ).

**Definition 2.** The **gap-ratio**,  $\Gamma_M(x)$ , at a location  $x$  relative to a point set  $M$ .  $\Gamma_M(x) = R(x)/r(x)$ , where  $R(x)$  is the radius of the largest disc containing  $x$  but not intersecting  $M \setminus x$ , and  $r(x)$  is the nearest neighbor of  $x$  in  $M \setminus x$ . At the boundary of our geometric domain, we require that  $x$  and the center of the largest disc be contained in the convex closure of  $M$ . For shorthand, we say a mesh  $M$  has gap-ratio quality  $\Gamma$  if  $\forall x, \Gamma_M(x) \leq \Gamma$ .

Note that if some mesh  $M$  has some constant gap-ratio quality  $\Gamma$ , then it has no arbitrarily small or large angles.

**Definition 3.** The **local feature size**,  $\text{lfs}_M(x)$ , at a point  $x$  relative to a PSLG  $M$  is given by the radius of the smallest disc centered at  $x$  that intersects two disjoint features (segments or vertices) of  $M$ . When  $M$  is suppressed,  $\text{lfs}(x)$  will be with reference to the input PSLG. Additionally,  $\text{lfs}_0(x)$  is the radius of an identically defined disc intersecting two vertices of the input, ignoring segments.

Given these definitions, we state our main result:

**Theorem 1.** On input a PSLG Algorithm Overlay Stitch Meshing generates a conforming triangulation  $\mathcal{T}$  with no angle greater than  $170^\circ$ . The size of

$|\mathcal{T}| \leq c \log(L/s) \cdot OPT$  where  $OPT$  is the size of an optimal conforming triangulation with all angle bound away from  $180^\circ$  and  $c$  is some fixed constant.

## 1.2 Related Work

The present work derives from two disjoint lines of past research, well-graded meshes and no-large-angle triangulation. Both lines of research trace their lineage back to the same motivating problem, that of producing quality meshes for finite-element simulations. However, the methods, guarantees, and final output provided by each field are drastically different.

Well-graded meshing algorithms attempt to produce meshes in which the length of each edge is proportional to the lfs at its end points. These algorithms generally fall into two categories, structured and unstructured, as typified by Quadtree methods [1] and Delaunay refinement methods [2, 4] respectively.

One significant property of most well-graded meshing algorithms is that they provide guarantees regarding both the largest and the smallest angles in the resulting mesh. A bound on the smallest angle is a sufficient but not necessary condition for a mesh to have no large angles. However, many Delaunay refinement algorithms are only guaranteed to terminate for inputs where all input angles are at least  $60^\circ$ . Several algorithms avoid this restriction by severely weakening the smallest output angle guarantee in order to guarantee termination [9].

Significant research has been done on how to extend well-graded meshing algorithms to efficiently handle small input angles, most of which is based on extending the “concentric shelling” method proposed by Ruppert in [2]. The main idea requires that if two edges share a vertex, then the splitting of these edges should be identical, effectively “protecting” the region inside the small angle. Extensions of the protective region approach to 3D have so far been quite involved, and output size guarantees fairly weak [10, 11, 12]. The immediate consequence of concentric shelling is that the size and grading of the output mesh depends on the smallest angle in the input PSLG.

Even in the absence of small input angles, well-graded meshing techniques often produce meshes whose size is linearly dependent on the spread ( $L/s$ ) of the underlying vertex input set. Generally, the spread is assumed to be some polynomial in the size of the input.

Because of the blowup in size associated with well-graded meshing, much research has focused on the problem of eliminating large angles. Bern, Dobkin, and Eppstein give an algorithm that produces a no-large-angle mesh of a simple polygon using  $O(n \log n)$  triangles; or  $O(n^{3/2})$  triangles for polygons with holes [13]. This result was later improved by Bern, Mitchell, and Ruppert who gave an algorithm that produces a nonobtuse triangulation of a polygon with holes using  $O(n)$  triangles [14]. For arbitrary planar straight line graphs, there is a lower bound attributed to Paterson that says  $\Omega(n^2)$  triangles may be necessary.

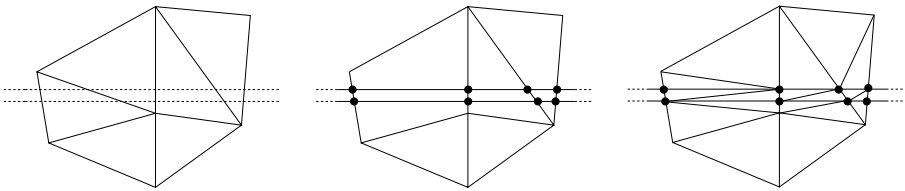
An algorithm of Mitchell [15] produces a triangulation with bounded angles and at most  $O(n^2 \log n)$  vertices. This was later improved by Tan [16] who improved the angle bounds to  $\frac{11}{15}\pi$  and the size bounds to  $O(n^2)$  vertices. The latter

result is within a constant factor of worst-case optimal. Both of these algorithms use the method of propagating paths in which a bad angle is subdivided, possibly creating a new large angle elsewhere. The methods and analysis depend on bounding the length of the paths of bad angles as they propagate through the triangulation. We present a much simpler algorithm that gives a triangulation with angles bounded below  $170^\circ$  of size  $O(\log L/s)$ -competitive with any triangulation achieving this angle bound. For inputs that admit a small no-large-angle triangulation, this improves on the worst-case optimal bound of  $O(n^2)$ . It is possible that propagating path algorithms also produce competitively sized meshes but a competitive analysis has proven difficult, even using the lower bound techniques of this paper.

The main technology used in OSM is a well-graded *overlay* mesh on just the input vertices. The input edges are then *stitched* in. The use of an axis-aligned overlay grid was introduced by Baker et al [17] and later extended by Dey [18]. The same paradigm can be traced in structured meshing algorithms that use a quadtree as an overlay and then stitch in the input edges [1]. In this paper, we use an unstructured triangular mesh as our overlay.

## 2 The OSM Algorithm

The algorithm has three phases: the Overlay mesh phase, the Stitching phase, and the Completion phase. In the first phase, a standard point set meshing algorithm is run on the input vertices to form the *overlay mesh*. In the second phase, the input segments are *stitched* into the mesh, carefully choosing to add vertices at some intersections of the overlay mesh and the input segments. The Stitching phase will leave some non-triangular faces. In the last phase, the Completion phase, these leftover faces are triangulated to minimize the largest angle. See Figure 1.



**Fig. 1.** The basic stages of overlay meshing: At left, an overlay mesh is generated that does not yet conform to input segments (shown dashed). Center, the input segments are stitched into the mesh by adding some intersections and discarding some overlay edges. At right, the triangulation is completed.

### 2.1 Phase 1: The Overlay Mesh

The overlay mesh is constructed on the input vertices using the Sparse Voronoi Refinement meshing algorithm, applied to point sets [19]. The output is a Delaunay mesh conforming to the input vertices. The overlay mesh has gap-ratio

quality  $\Gamma$ , for a constant parameter  $\Gamma > 1$ . No angle in the overlay mesh will be smaller than  $\theta_1 = \arcsin(1/2\Gamma)$ . The size of the mesh (in terms of vertices) will be guaranteed to be  $O\left(\int_{\Omega} \frac{1}{(\text{fs}_0(z))^2} dz\right)$ .

The choice of SVR as a meshing algorithm for the overlay mesh is not necessarily critical for Overlay Stitch Meshing. SVR is used because of the strong runtime guarantees and the tightest analyzed bounds on  $\Gamma$  when applied to point sets. Any point set meshing algorithm could be used, so long as it outputs in a Delaunay mesh with guarantees on  $\Gamma$  and output size. The usual method is to set  $\Gamma = 1 + \varepsilon$  for some small constant  $\varepsilon$ , yielding a  $\theta_1$  slightly less than  $30^\circ$ .

## 2.2 Phase 2: Stitching in Edges

We now wish to begin conforming to the input segments. We can look at all the intersections between overlay edges and input segments, and we will classify every intersection as good or bad. A good intersection is approximately perpendicular, meaning that the segment and overlay edge meet at angles larger than  $\theta_1$ . A bad intersection is approximately parallel (angles smaller than  $\theta_1$ ).

If an overlay edge crosses no segment, or crosses any segment with a good intersection, it will be kept. Overlay edges that cross segments at solely bad intersections will be discarded. The intuition here is that if an edge of the overlay mesh intersects input segments only in a parallel fashion, then we can use the input segments instead, so we throw out the overlay edge.

We have kept some overlay edges that cross input segments. We will then add Steiner points that subdivide input segments where they intersect these crossing edges (at good or bad intersections). We also now add all the subdivided input segments to the mesh. The result is not, in general, a triangulation, although it does now conform to the input.

### Stitching in edges

```

1: for all input segments  $e$  do
2:   Compute intersections of  $e$  with overlay mesh edges
3:   if some overlay edge  $e'$  intersects  $e$  at an angle greater than  $\theta_1$  then
4:     mark  $e'$  as kept.
5:   /* Insert the 'good' intersections */
6:   for all intersections do
7:     if the overlay edge at the intersection is marked as kept then
8:       insert the intersection point into the mesh splitting corresponding input segment.
9:     else
10:      Remove the overlay edge at the intersection from the mesh.
11:   /* Recover the input */
12:   for all input subsegments  $e$  do
13:     insert  $e$  into the mesh

```

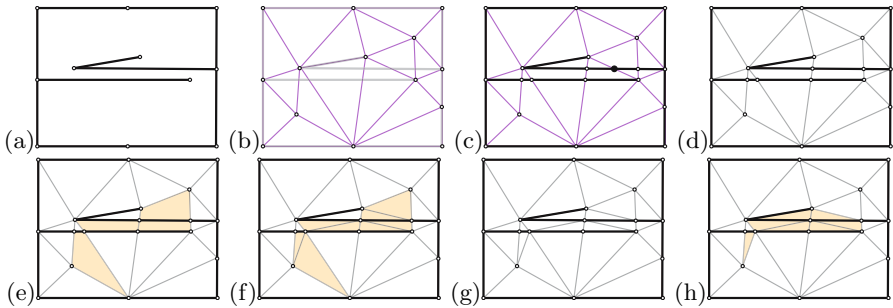
## 2.3 Phase 3: Completing the Mesh

After the edges are stitched in, all that remains is to add enough edges to get back to a triangulation. Each remaining non-triangular face is triangulated to

minimize the maximum angle. Lemma 1 implies that this last step can naively be done efficiently because all of the non-triangular faces have at most six sides. Theorem 2 will show that these faces can always be triangulated to have no large angles.

**Lemma 1.** *After the stitching phase, all faces have at most 6 sides.*

A rigorous proof of Lemma 1 would proceed by first considering the case where no overlay edges are discarded. Then we can inductively remove the discarded edges, preserving the bound of six sides to every face.



**Fig. 2.** A sample run of OSM. In box (a), an input PSLG is given. The input contains both a small angle and a narrow pair of edges. In (b), the first phase of OSM constructs a quality overlay mesh on the input points, ignoring the input segments, and adding Steiner points where necessary. In (c), the input is intersected with the overlay mesh and one bad intersection is identified. Box (d) shows the (not yet triangular) mesh after the stitching phase. The remaining non-triangular faces highlighted in (e) are triangulated in the completion phase (f). The final mesh is shown in (g) and the remaining small angle triangles are shown in (h).

### 3 Output Angle Guarantees

The purpose of this section is to prove that the maximum angle in the output mesh is bounded from above by a constant  $\pi - \theta_2$  that is independent of the smallest input angle. Here,  $\theta_2$  is a constant depending only on  $\theta_1$ , the smallest angle in the overlay mesh. This result is stated formally below as Theorem 2. If our overlay mesh has  $\Gamma = 1 + \epsilon$ , then we find that  $\theta_1 \geq 29.9^\circ$ , and  $\theta_2 = \arcsin(1/(4 + 2 \cos \theta_1 + O(\epsilon))) \geq 10^\circ$ . First, we prove an important lemma regarding the angles at vertices inserted during the stitching phase.

**Lemma 2.** *If an overlay mesh edge  $e$  intersects an input segment  $f_1$  at an angle greater than  $\theta_1$  then no input segment  $f_2$  can intersect  $e$  at an angle less than  $\theta_2 = \arcsin\left(\frac{1}{2\Gamma(2\Gamma + \cos \theta_1)}\right)$ .*

*Proof.* Let  $F_1$  and  $F_2$  be the lines containing edges  $f_1$  and  $f_2$  respectively, and let  $v = F_1 \cap F_2$ . The statement is trivial if  $F_1, F_2$  are parallel or identical, so we may assume that  $v$  is a single point.

Without loss of generality, let us assume that the edge  $e$  is horizontal and  $v$  lies below the line containing  $e$ . Also, by symmetry, we may assume that the small angle intersections we are worried about are on the left.

Figure 3 (Left) shows the edge  $e$  with the circumcircle  $C_1$  of a triangle containing it. The gap ratio guarantees that there is an empty ball  $C_2$  centered at  $a$  with radius at least  $1/\Gamma$  times the radius of  $C_1$ . The point  $d$  is the lower intersection of these two circles. In the figure  $\theta_1 = \angle abd$ . The line  $L_2$  contains  $\overline{bd}$  and the line  $L_1$  is parallel to  $L_2$  passing through  $a$ . The point  $c$  is the lower intersection of  $L_1$  and  $C_2$ .

Since  $f_1$  passes through  $e$  at an angle greater than  $\theta_1$ ,  $v$  must lie below  $L_1$ . In order for  $f_2$  to intersect  $e$  at an angle less than  $\theta_1$ ,  $v$  must lie above  $L_2$ . Thus we see that the intersection  $v$  must lie somewhere in the shaded region. Therefore the smallest possible angle between  $f_2$  and  $e$  occurs if  $v = c$  and the angle is:

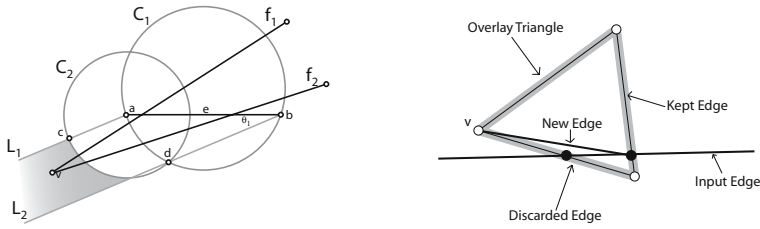
$$\angle abc \geq \arcsin\left(\frac{\sin \theta_1}{2\Gamma + \cos \theta_1}\right) = \arcsin\left(\frac{1}{2\Gamma(2\Gamma + \cos \theta_1)}\right)$$

The preceding Lemma ensures that if OSM chooses not to discard an edge of the overlay mesh, then that edge will not create any large angles. We can now proceed to the main theorem about the output angle guarantees.

**Theorem 2.** *All angles in the final mesh are bounded from above by  $\pi - \theta_2$ , where  $\theta_2 = \arcsin\left(\frac{1}{2\Gamma(2\Gamma + \cos \theta_1)}\right)$  is the lower bound on the angle of intersection between a kept edge of the overlay mesh and an input edge. For  $\Gamma = 1 + \varepsilon$ , this gives a largest angle of at most  $170^\circ$ .*

*Proof.* We need to consider the angles at two types of vertices, those vertices that appear in the overlay mesh and those that are added during the stitching phase. In the latter case, we were careful only to add vertices when the angles of intersection met this criterion. Thus, although we may have added more edges incident to such vertices in order to get a triangulation, the maximum angle at such vertices clearly achieves the desired bound. In the case of a vertex  $v$  from the overlay mesh, we have to be a little more careful. The overlay mesh itself satisfied this property so all large angles at overlay vertices must arise from discarded overlay edges. The completion phase of the algorithm returns the min-max angle triangulation of the non-triangular phases. It will suffice to show that there exists some triangulation that guarantees no large angles.

Observe that any input edge  $e$  crossing a triangle  $t$  of the overlay mesh can cause at most one edge of  $t$  to be discarded. This is because  $\theta_1 = \arcsin\left(\frac{1}{2\Gamma}\right)$  is a lower bound on the smallest angle in the overlay mesh, and thus the largest angle is at most  $\pi - 2\theta_1$ . It follows that at least one of these edges must intersect  $e$  at an angle greater than  $\theta_1$  and therefore will not be discarded. This fact implies that when an edge gets discarded, we can replace it with one that has been rotated by at most  $\theta_1$ . This is illustrated in Figure 3 (Right). The resulting



**Fig. 3.** *Left.* Circumball  $C_1$  and gap radius ball  $C_2$  for an edge  $e$  and its endpoint  $a$  respectively. The lines  $L_1$  and  $L_2$  are parallel. The point  $v$  must be in the shaded region. *Right.* The input edge causes exactly one edge of the overlay triangle to be discarded and replaced with a new edge whose angle with the discarded edge is less than  $\theta_1$ . The angles at  $v$  are changed by at most  $\theta_1$ .

largest angle at  $v$  is at most  $\pi - \theta_1$ . Now, if we look at the edges ordered radially around  $v$ , we see that no two adjacent edges can be rotated apart from each other. This is because if two edge of the triangle  $t$  are discarded then they both get replaced with edges that terminate on the third edge and thus lie entirely within the triangle. Thus the angle at  $v$  is strictly smaller in this case.

### 4 Size of the Triangulation

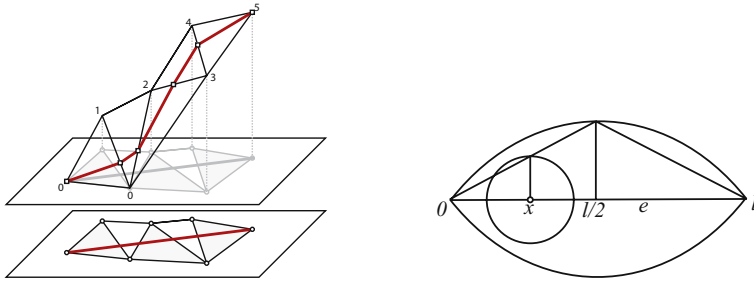
In this section we show that the size of the output mesh is determined only by the local feature size of the input PSLG.

**Lemma 3.** *For any input edge  $e$ , the number of triangles in the overlay mesh intersecting  $e$  is*

$$O\left(\int_{z \in e} \frac{1}{\text{lfs}_0(z)} dz\right). \tag{1}$$

*Proof.* Let  $t_1, \dots, t_k$  be a minimal ordered sequence of adjacent triangles in the overlay mesh that covers  $e$ . Assign heights to the vertices of  $t_2, \dots, t_k$  so that the highest point of  $t_i$  is one more than the highest point of  $t_{i-1}$ . Set the heights for  $t_1$  to be  $0, 0, 1$ . Let  $e_i = e \cap t_i$  be the subsegment of  $e$  contained in triangle  $t_i$ . We consider the lifted version of  $e$ , call it  $e^+$ , to be the polygonal chain in  $\mathbb{R}^3$  on the surface of these lifted triangles whose segments project down onto  $e$ .

Observe that the gradient of a lifted triangle  $t$  cannot be too steep because both the smallest angle of  $t$  and the maximum height difference between vertices of  $t$  are bounded by constants. The maximum difference between the height of the vertices of a triangle is bounded by the degree of vertices. Thus the gradient of  $t$  is bounded by  $\frac{\gamma}{r}$  where  $r$  is the radius of the circumcircle of  $t$  and  $\gamma > 0$  is a constant. Here, the constant  $\gamma$  depends only on the gap ratio  $\Gamma$  of the overlay mesh and the maximum degree of any vertex in the overlay mesh which is also a constant because the overlay mesh has small angle guarantees.



**Fig. 4. Left.** An edge cuts through the overlay triangulation. Heights are assigned to the vertices so that the maximum height is exactly the number of triangles cut. **Right** An edge with an empty lens around it. The radius of the circle centered at  $x$  is the height of the triangle at  $x$ . This guarantees that the circle is contained entirely within the lens.

Let  $e_i$  be the subsegment of  $e$  lying in triangle  $t_i$ . The change in height along  $e_i^+$  is at most  $|e_i| \frac{\gamma}{r_i}$  where  $r_i$  is the circumradius of triangle  $t_i$ . Thus, the total change in height  $k$  along  $e^+$  is bounded as follows.

$$k \leq \sum_{i=1}^k |e_i| \frac{\gamma}{r_i} \leq \gamma \sum_{i=1}^k \int_{e_i} \frac{1}{r_i} dx \tag{2}$$

The triangles  $t_i$  are part of a well graded mesh and therefore,  $\text{lfs}_0(x) = cr_i$  for all  $x \in t_i$  and some constant  $c$ . So, we can rewrite the above inequality as follows to complete the proof.

$$k \leq \gamma \sum_{i=1}^k \int_{e_i} \frac{c}{\text{lfs}_0(x)} dx = c\gamma \int_e \frac{1}{\text{lfs}_0(x)} dx \tag{3}$$

**Theorem 3.** *The number of Steiner points added during the course of the algorithm is*

$$O \left( \int_{\Omega} \frac{1}{(\text{lfs}_0(z))^2} dz + \int_E \frac{1}{\text{lfs}_0(z)} dz \right). \tag{4}$$

where  $E$  is the set of input edges and  $\Omega$  is the input domain (i.e. the plane).

*Proof.* We look at the two phases of OSM where vertices are added. First, in the construction of the overlay mesh, the number of points added is  $O \left( \int_{\Omega} \frac{1}{(\text{lfs}_0(z))^2} dz \right)$ , as guaranteed by the SVR algorithm for point set meshing [19].

Second, in the stitching phase, we choose a subset of the intersections along each edge with the overlay mesh. It follows from Lemma 3 that the total number of intersections is  $O \left( \int_E \frac{1}{\text{lfs}_0(z)} dz \right)$ . Therefore, it follows that the subset of these that we keep also achieves this bound. The statement of the theorem follows directly from summing the Steiner points added in each phase.



### 4.1 Competitive Results

The  $\alpha$ -lens is the main tool we will use to analyze the optimal mesh for a given maximum angle guarantee. Recall that an  $\alpha$ -lens on a line segment  $\overline{xy}$  is the set of all points  $z$  such that  $\angle xzy \geq \alpha$ . A lens is the intersection of two disks with the same radius. We note one important fact about  $\alpha$ -lenses: If  $\overline{ab}$  is a subsegment of  $\overline{xy}$  then the  $\alpha$ -lens around  $\overline{ab}$  is strictly contained in the  $\alpha$ -lens around  $\overline{xy}$ .

**Theorem 4.** *The output of OSM is a mesh with no large angles that is at most  $O(\log(L/s))$  times the size of any mesh achieving the same maximum angle guarantee for some fixed constant  $c$ .*

*Proof.* Just as in Theorem 3, we will consider the Steiner points added during the Overlay phase separate from those added during the Stitching phase. The Overlay phase only adds  $O(n \log(L/s))$  and any mesh conforming to the input has size  $\Omega(n)$  so we need only worry about Steiner points added during the Stitching phase.

Suppose we have an optimal size mesh  $M_{OPT}$  with the property that no angle is greater than some constant  $\alpha$ . For any edge  $e$  in  $M_{OPT}$ , the  $\alpha$ -lens around  $e$  contains no other vertices of  $M_{OPT}$ . This is just another way of stating the no large angle property. In particular, the edges  $e$  that are subsegments of input edges have  $\alpha$ -lenses that contain no input vertices.

In order to prove that OSM is  $\log(L/s)$ -competitive, it will suffice to prove that OSM stitches at most  $\log(L/s)$  Steiner points on any input subsegment of  $M_{OPT}$ . Recall that  $\text{lfs}_0(x)$  is the radius of the smallest circle centered at  $x$  containing two input vertices. In general,  $\text{lfs}_0$  is lower bounded by  $s$ , the distance between the two closest vertices in the mesh. We get a better lower bound on  $\text{lfs}_0$  near input edges from the fact that the lenses around the input subsegments of  $M_{OPT}$  contain no input vertices. We will use both of these lower bounds on  $\text{lfs}_0$  to upper bound the integral from Theorem 3. Recall that this integral bounds the number of stitch vertices added on any edge.

For a particular input subsegment  $e$  in  $M_{OPT}$ , we parameterize  $e$  on the interval  $[0, l]$  where  $l$  is the length of  $e$  as in Figure 4. To compute the lower bound on  $\text{lfs}_0(x)$  for  $x \in [0, l/2]$  it suffices to show that there is an circle centered at  $x$  that contains no input vertices. We first inscribe an isosceles triangle into the top half of the  $\alpha$ -lens around  $e$ . The altitude of the triangle at  $x$  is  $\tan^{-1}(\alpha/2)x$ .

Consider the circle  $C$  centered at  $x$  with radius  $\tan^{-1}(\alpha/2)x$ . Observe that the top edge of the inscribed triangle cuts off some half  $\alpha'$ -lens from the circle  $C$  and some half  $\alpha''$ -lens from the original lens around  $e$ . Observe that  $\alpha' = \alpha''$  and thus the smaller lens is entirely contained in the larger. It follows that the circle  $C$  is contained entirely within the  $\alpha$ -lens around  $e$  and thus,  $C$  contains no input vertices. Therefore,  $\text{lfs}_0(x) \geq \tan^{-1}(\alpha/2)x$  for all  $x \in e$ .

We can now use our bound on  $\text{lfs}_0$  to bound the size integral from Theorem 3 as follows.

$$\int_0^l \frac{1}{\text{lfs}_0(x)} dx \leq 2 \left( \int_0^s \frac{1}{s} dx + \int_s^{\frac{l}{2}} \frac{1}{\text{lfs}_0(x)} dx \right) \leq 2 + 2 \tan \alpha/2 \int_s^{\frac{l}{2}} \frac{1}{x} dx \in O(\log \frac{L}{s})$$

## 5 Conclusions

### 5.1 Size Bounds in Terms of $n$

To better understand the size guarantees of OSM in relation to previous results that only analyze worst case performance, we can compute two coarse upper bounds on the mesh size.

First, we see that the mesh size is  $O(n^2 \log(L/s))$ . This follows from the fact that the overlay mesh is of size  $O(n \log(L/s))$ . There are at most  $O(n)$  edges so the stitching phase adds at most  $O(n^2 \log(L/s))$  points. When  $L/s \in O(\text{poly}(n))$ , this bound exactly matches the  $O(n^2 \log n)$  of Mitchell on triangulating with no large angles from [15].

Alternatively, one could recompute the sizing integral from Theorem 3 using  $s$  as a lower bound on  $\text{lfs}_0$ . The result is an  $O(n(L/s))$  upper bound on the mesh size. So, when  $L/s \in O(n)$ , the output size is  $O(n^2)$ , matching the bound of Tan [16]. Certain pathological examples such as Paterson's example (see [13]) requiring  $\Omega(n^2)$  Steiner points can be drawn so that  $L/s \in O(n)$ . Thus, OSM is worst case optimal when the input vertices have linear spread.

For reasonable inputs where  $L/s \in O(\text{poly}(n))$ , Theorem 4 implies that the output of OSM is  $O(\log n)$ -competitive. For inputs that admit  $O(n)$  no-large-angle meshes, this is a factor of  $n/\log n$  better than the Tan guarantee [16].

### 5.2 Work Efficiency

Overlay Stitch Meshing can be easily implemented to run in time and space  $O(n \log(L/s) + m)$ . The runtime of the SVR algorithm used in the overlay phase was analyzed in [19]. This stage is the majority of the work. Simple arguments can show that the stitching and completion phases can be implemented as  $O(m)$  post-processes. In the case of input with  $O(\text{poly}(n))$  spread, this is asymptotically optimal work.

### 5.3 Extensions

The most obvious extension is into three and higher dimensions. Obtaining good guarantees on output size for meshing algorithms in three dimensions remains an interesting open problem [10, 11, 12]. For the general input case, unstructured meshing research has focused on concentric shelling techniques that have complicated implementations and have yielded no strong sizing results.

## References

- [1] Bern, M., Eppstein, D., Gilbert, J.R.: Provably Good Mesh Generation. *Journal of Computer and System Sciences* 48(3), 384–409 (1994)
- [2] Ruppert, J.: A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms* 18(3), 548–585 (1995)
- [3] Mitchell, S., Vavasis, S.: Quality mesh generation in three dimensions. In: *Proc. 8th ACM Symp. Comp. Geom.*, pp. 212–221. ACM Press, New York (1992)

- [4] Shewchuk, J.R.: Delaunay Refinement Algorithms for Triangular Mesh Generation. *Computational Geometry: Theory and Applications* 22(1–3), 21–74 (2002)
- [5] Babuška, I., Aziz, A.K.: On the Angle Condition in the Finite Element Method. *SIAM Journal on Numerical Analysis* 13(2), 214–226 (1976)
- [6] Guattery, S., Miller, G.L., Walkington, N.: Estimating interpolation error: A combinatorial approach. In: *Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, ACM and SIAM, pp. 406–413 (1999)
- [7] Boman, E., Hendrickson, B., Vavasis, S.: Solving elliptic finite element systems in near-linear time with support preconditioners (2004)
- [8] Miller, G., Vavasis, S.: Only large angles mater. Private communications (2005)
- [9] Pav, S.E.: Delaunay Refinement Algorithms. PhD thesis, Department of Mathematics, Carnegie Mellon University, Pittsburgh, Pennsylvania (2003)
- [10] Cheng, S.-W., Poon, S.-H.: Graded Conforming Delaunay Tetrahedralization with Bounded Radius-Edge Ratio. In: *Proceedings of the Fourteenth Annual Symposium on Discrete Algorithms*, Baltimore, Maryland. Society for Industrial and Applied Mathematics, pp. 295–304 (2003)
- [11] Pav, S.E., Walkington, N.J.: Robust Three Dimensional Delaunay Refinement. In: *Thirteenth International Meshing Roundtable*, Williamsburg, Virginia, Sandia National Laboratories, pp. 145–156(2004)
- [12] Cheng, S.W., Dey, T.K., Ramos, E.A., Ray, T.: Quality Meshing for Polyhedra with Small Angles. In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, Brooklyn, New York. Association for Computing Machinery, pp. 290–299 (2004)
- [13] Bern, M.W., Dobkin, D.P., Eppstein, D.: Triangulating polygons without large angles. *Int. J. Comput. Geometry Appl.* 5, 171–192 (1995)
- [14] Bern, M.W., Mitchell, S.A., Ruppert, J.: Linear-size nonobtuse triangulation of polygons. *Discrete & Computational Geometry* 14(4), 411–428 (1995)
- [15] Mitchell, S.A.: Refining a triangulation of a planar straight-line graph to eliminate large angles. In: *34th Annual Symposium on Foundations of Computer Science*, Palo Alto, California, pp. 583–591. IEEE, Los Alamitos (1993)
- [16] Tan, T.S.: An optimal bound for high-quality conforming triangulations. *GEOMETRY: Discrete and Computational Geometry* 15 (1996)
- [17] Baker, B.S., Grosse, E., Rafferty, C.S.: Nonobtuse triangulation of polygons. *Discrete Comput. Geom.* 3(2), 147–168 (1988)
- [18] Dey, T.K.: Good triangulations in the plane. In: *CCCG: Canadian Conference in Computational Geometry* (1990)
- [19] Hudson, B., Miller, G., Phillips, T.: Sparse Voronoi Refinement, Technical Report CMU-CS-06-132, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania (2006)

# A Fully Abstract Trace Semantics for General References

J. Laird\*

Dept. of Informatics, University of Sussex, UK  
jim1@sussex.ac.uk

**Abstract.** We describe a fully abstract trace semantics for a functional language with locally declared general references (a fragment of Standard ML). It is based on a bipartite LTS in which states alternate between program and environment configurations and labels carry only (sets of) basic values, location and pointer names. Interaction between programs and environments is either direct (initiating or terminating subprocedures) or indirect (by the overwriting of shared locations): actions reflect this by carrying updates to the shared part of the store.

The trace-sets of programs and contexts may be viewed as deterministic strategies and counter-strategies in the sense of game semantics: we prove soundness of the semantics by showing that the evaluation of a program in an environment tracks the interaction between the corresponding strategies. We establish full abstraction by proving a definability result: every bounded deterministic strategy of a given type is the trace-set of a configuration of that type.

## 1 Introduction

The conjunction of functional programming and general references is a powerful one — for example, it can describe both object-oriented and aspect-oriented [\[1\]](#) computation by translation. So it is not, perhaps, surprising that the behaviour of functional programs with locally bound references is difficult to reason about; they may exhibit a variety of subtle phenomena such as aliasing, and self-referencing and self-updating variables. In some respects, the higher-order “pointer-passing” exhibited by such programs is analogous to process-passing in a concurrent setting. The most significant differences between pointer-passing and process-passing are that the former typically takes place sequentially, between a program and its environment rather than between processes in parallel, and that pointers and locations may be passed through the medium of the store, where they persist until overwritten. In this paper, we describe a labelled transition system for a sequential functional language with general references which captures these aspects of reference passing, and use it to give a sound, complete and direct characterization of contextual equivalence.

---

\* Supported by UK EPSRC grant GR/S72181.

Another objective of this research is develop the correspondence between labelled transition systems and another, more abstract form of trace-based representation of references. *Game semantics* has been used to give a denotational model [1] of essentially the same functional language with general references as the the one studied here. The principal difference is that the language in [1] includes “bad references” — objects of reference type which do not behave as location names — due to the encoding of imperative variables as read-write methods rather than pointers. Because of this it does not precisely characterize the way names may be passed between functions and used to interact via the store in languages like Standard ML with only good variables. By passing location names directly, we are able to capture interaction via the shared part of the store, yielding a full abstraction result for such a language.

The languages modelled here and in [1] are equivalent when restricted to terms of purely functional type (which in any case suffice to represent bad variables). We first present a basic trace semantics which is sound and complete for this fragment, and in which there is a simple correspondence between actions over a given type, and moves in the corresponding game, which extends to relate ( $\alpha$ -equivalence classes of) traces to justified, well-bracketed sequences (using pointer-names to determine justification pointers) and deterministic trace-sets to strategies. We prove a game-semantics-style “definability result”, showing that every bounded “deterministic strategy” is the trace-set of a term.

The full LTS is based on the same set of actions and so is a conservative extension of the basic one as far as the functional fragment is concerned. It accounts for interaction through the store by adding an update of every shared location (as a pointer or atomic value) to each action. The soundness, definability and full abstraction results extend straightforwardly to this setting.

Labelled transition systems for similar functional languages have been presented, including one for a language with (global) general references [6]. In this system, labels carry the contexts which are used to test the term; in the one describe here, labels are more basic, containing only (sets of) atomic values, and so the resulting trace semantics could be said to be more abstract. In this respect it resembles the triggering semantics of the higher-order  $\pi$ -calculus [10], which motivated its development. There are close parallels with the trace semantics of a core fragment of Java presented in [7], including the use of a merge operation on context stacks.

We may compare labelled transition systems with various approaches to reasoning about contextual equivalence in functional-imperative languages, such as the bisimulation based methods of Koutavas and Wand [8], which are sound and complete for an (untyped) language similar to the one described here, or the Hoare-logic style approach of Honda, Berger and Yoshida [3]. These could be seen as characterizing equivalence using a more restricted set of contexts, but trace semantics actually gives an explicit description of the (observationally relevant) behaviour of a term.

The correspondence between game semantics and pointer-based evaluation of functional languages was first investigated in [4], where the interaction between

the strategies denoting two terms of the  $\lambda$ -calculus was shown to corespond to the “run” of the Krivine abstract machine evaluating the application of one to the other. This is very similar to the proof of a result used here to establish that trace inclusion is a precongruence: that the result of evaluating a program in a given environment is determined by the “parallel composition” of their traces. Another route to a trace-based representation of functional programs is by representing them as terms of the  $\pi$ -calculus, for which a sequential fragment has been identified by typing [2] and the connections with game semantics examined in [5].

## 2 A Functional Language with General References

The language we study,  $\mathcal{L}$ , is a small fragment of Standard ML. It is essentially the language modelled in [1], without bad variables (and restricted to the ground types `com` and `bool`, for simplicity) — a simply-typed functional language with assignment, dereferencing and declaration of general references. Thus types are given by the grammar:

$$T ::= \text{com} \mid \text{bool} \mid T \Rightarrow T \mid \text{var}[T]$$

Terms are those of the simply-typed  $\lambda$ -calculus over these types, with the following constants:

- Assignment**  $\text{assign}_T : \text{var}[T] \Rightarrow T \Rightarrow \text{com}$
- Dereferencing**  $!_T : \text{var}[T] \Rightarrow T$
- Variable declaration**  $\text{ref}_T : T \Rightarrow \text{var}[T]$
- Conditional**  $\text{If}_T : \text{bool} \Rightarrow T \Rightarrow T \Rightarrow T$
- Equality Testing**  $\text{eq}_T : \text{var}[T] \Rightarrow \text{var}[T] \Rightarrow \text{bool}$
- Basic Values**  $() : \text{com}$  and  $\text{tt}, \text{ff} : \text{bool}$

We assume that each variable has a unique type, so that any well-formed term  $M$  has a unique type  $\text{ty}(M)$  (although we shall generally omit typing annotations where they are not necessary). We sugar the language by writing  $M := N$  for  $(\text{assign } M) N$ ,  $M; N$  for  $(\lambda x.N) M$  with  $x$  not free in  $N$ ,  $\text{new } x := M.N$  for  $(\lambda x.N) (\text{ref } M)$  (or just  $\text{new } x.N$  if  $M$  is an arbitrary value, which we may select at any type),  $\Omega$  for  $(\text{new } x.x := \lambda y.(!x ()) !x) ()$ , and  $M = N$  for  $(\text{eq } M) N$ .

A key distinction will be between functional values (i.e. values of function type), which are  $\lambda$ -abstractions, function-constants and pointer names (variables of function-type) — and non-functional values, which are location names (variables of reference type) and the basic values  $()$ ,  $\text{tt}$ ,  $\text{ff}$ . We mark this distinction by using  $F, G$  for functional values and  $v, u, w$  for non-functional values; the two notions of value are given by the following grammars:

$$F, G ::= x \mid \lambda x.N \mid ! \mid \text{If} \mid \text{eq} \mid \text{assign} \mid (\text{eq } v) \mid \text{assign } v$$

$$v, u ::= a \mid () \mid \text{tt} \mid \text{ff}$$

where  $x$  ranges over pointer names and  $a$  over location names. We write  $V, U$  for values of arbitrary type. *Atomic* values, for which we write  $\phi, \psi$ , are basic values, pointer or location names.

Our LTS for  $\mathcal{L}$  is based on a small-step operational semantics (Table [III](#)). Terms are evaluated in an environment consisting of a set of pointers  $\mathcal{B}$  — a finite partial function from pointer names to functional values and a store  $\mathcal{S}$  — a finite partial function from location names to values. Non-functional values may be substituted directly, but substitution of a functional value is via a freshly generated pointer name bound to it. The reduction rules are based on typed *evaluation contexts* given by the following grammar:

$$E[\_]_T := [\_]_T \mid F E[\_]_T \mid E[\_]_T M$$

By typing the “hole”, we ensure that any typable context  $E_T[\_]$  has a unique type  $\text{ty}(E_T[\_])$ . Throughout, we adopt the convention that in any rule  $C \longrightarrow C'$ , any variable mentioned explicitly in  $C'$  which is not mentioned explicitly in  $C$  is assumed to be fresh — i.e. not occurring free or bound in  $C$ . We write  $M \Downarrow$  if

**Table 1.** Reduction Rules for Program Evaluation

|  |  |
|--|--|
| $E[(\lambda x.M) v]; \mathcal{B}; \mathcal{S} \longrightarrow E[M[v/x]]; \mathcal{B}; \mathcal{S}$               | $E[(\lambda x.M) F]; \mathcal{B}; \mathcal{S} \longrightarrow E[M[y/x]]; \mathcal{B}, (y, F); \mathcal{S}$       |
| $E[x V]; \mathcal{B}, (x, F); \mathcal{S} \longrightarrow E[F V]; \mathcal{B}, (x, F); \mathcal{S}$              | $E[\text{ref } V]; \mathcal{B}; \mathcal{S} \longrightarrow E[a]; \mathcal{B}; \mathcal{S}, (a, V)$              |
| $E[!l]; \mathcal{B}; \mathcal{S}, (l, V) \longrightarrow E[V]; \mathcal{B}; \mathcal{S}, (l, V)$                 | $E[l := U]; \mathcal{B}; \mathcal{S}, (l, V) \longrightarrow E[()]; \mathcal{B}; \mathcal{S}, (l, U)$            |
| $E[l = l]; \mathcal{B}; \mathcal{S} \longrightarrow E[\text{tt}]; \mathcal{B}; \mathcal{S}$                      | $E[l = m]; \mathcal{B}; \mathcal{S} \longrightarrow E[\text{ff}]; \mathcal{B}; \mathcal{S}$                      |
| $E[\text{If tt}]; \mathcal{B}; \mathcal{S} \longrightarrow E[\lambda x. \lambda y. x]; \mathcal{B}; \mathcal{S}$ | $E[\text{If ff}]; \mathcal{B}; \mathcal{S} \longrightarrow E[\lambda x. \lambda y. y]; \mathcal{B}; \mathcal{S}$ |

$M; \_; \_ \longrightarrow^* ()$ ;  $\mathcal{B}; \mathcal{S}$  for some  $\mathcal{B}, \mathcal{S}$ . We define standard notions of observational approximation and equivalence:  $M \lesssim N$  if  $C[M] \Downarrow$  implies  $C[N] \Downarrow$  for all compatible closing contexts  $C[\_] : \text{com}$ . Our full abstraction result is given for closed terms, although the extension to terms with free variables is straightforward — note that  $M \lesssim N$  if and only if  $\lambda x.M \lesssim \lambda x.N$ . We may prove the following Context Lemma using standard techniques.

**Lemma 1.** *For any closed  $M, N : T$ ,  $M \lesssim N$  if for every evaluation context  $E[\_]_T : \text{com}$ , and store  $\mathcal{S}$ ,  $E[M]; \_; \mathcal{S} \Downarrow$  implies  $E[N]; \_; \mathcal{S} \Downarrow$ .*

### 3 Trace Semantics

We shall develop our trace semantics for  $\mathcal{L}$  by first giving a basic LTS, and proving that it yields a sound and complete model of the functional fragment of  $\mathcal{L}$  — i.e. for terms of type  $T$ , where  $T$  contains no reference type as a subtype. We shall then extend the LTS by adding a “shared store” component to its actions, and show that the resulting semantics is sound and complete for all terms of  $\mathcal{L}$ . This two-stage presentation of the trace semantics allows us to capture two aspects of behaviour in  $\mathcal{L}$  separately — first, the control-flow between programs at function types, and then shared access to the store. It also allows a clearer perspective on the correspondence with the game semantics of general references. Note that we

can give a conservative translation of  $\mathcal{L}$  (except for equality testing on reference names) into the functional fragment by representing the references as objects of functional type with the correct assignment/dereferencing behaviour (this is how references are represented in the games model [II](#)): for example we may replace  $\mathbf{var}[T]$  with the type  $\mathbf{bool} \Rightarrow T \Rightarrow T$  and give the following macros for assignment, dereferencing and declaration.

- $\mathbf{assign} \mapsto \lambda x.\lambda y.((x \mathbf{tt}) y); ()$
- $! \mapsto \lambda y.((y \mathbf{ff}) c)$  (where  $c$  is an arbitrary value)
- $\mathbf{ref} \mapsto \lambda x.\mathbf{new} \ n := x.\lambda y.\lambda z.\mathbf{If} \ y \ \mathbf{then} \ (n := z); z \ \mathbf{else} \ !n$

The translation becomes fully abstract if we include a bad variable constructor  $\mathbf{mkvar} : (\mathbf{bool} \Rightarrow T \Rightarrow T) \Rightarrow \mathbf{var}[T]$  with the appropriate operational behaviour making  $\mathbf{var}[T]$  a retract of  $\mathbf{bool} \Rightarrow T \Rightarrow T$ .

The trace semantics is given by a bipartite LTS in which nodes are partitioned between *program-configurations* and *environment-configurations*. The former are tuples  $(M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E})$  consisting of a program  $M$ , internal pointers  $B$  and store  $\mathcal{S}$ , a set of “output pointers”  $\mathcal{O}$ , a set of typed “input pointer” names  $\mathcal{I}$  and an “execution stack”  $\mathcal{E}$  of evaluation contexts, the sets  $\mathbf{dom}(\mathcal{B}), \mathbf{dom}(\mathcal{O}), \mathcal{I}$  being pairwise disjoint. An environment-configuration  $E$  is a tuple  $(\mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E})$  — there is no program.

The type of the program configuration  $(M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E})$  is  $(\mathbf{ty}(M); \mathbf{dom}(\mathcal{O}); \mathcal{I}; \mathbf{ty}(\mathcal{E}))$ , where  $\mathbf{ty}(\mathcal{E})$  is the sequence of hole and result types of the evaluation contexts on  $\mathcal{E}$  — i.e.  $\mathbf{ty}(E_1[\_]\_{S_1}) : \dots : \mathbf{ty}(E_n[\_]\_{S_n}) = S_1 \cdot \mathbf{ty}(E_1[\_]\_{S_1}) \cdot \dots \cdot S_n \cdot \mathbf{ty}(E_n[\_]\_{S_n})$ . The type of an environment configuration  $(\mathcal{B}, \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E})$  is the tuple  $(\mathbf{dom}(\mathcal{O}), \mathcal{I}, \mathbf{ty}(\mathcal{E}))$ . A configuration is functional if none of its subtypes is a reference type.

The actions of the LTS are either unlabelled (silent) or labelled. The unlabelled actions are the reductions of the small-step semantics — i.e.:

$$\frac{M; \mathcal{B}; \mathcal{S} \longrightarrow M'; \mathcal{B}'; \mathcal{S}'}{M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E} \longrightarrow M'; \mathcal{B}'; \mathcal{S}'; \mathcal{O}; \mathcal{I}; \mathcal{E}}$$

Labelled actions come in complementary pairs  $\alpha, \bar{\alpha}$  (an input action and an output action). We write  $s^\perp$  for the trace in which each output action is swapped for its corresponding input and vice-versa. Input actions send  $E$ -configurations to  $P$ -configurations and output actions send  $P$ -configurations to  $E$ -configurations and so traces alternate between them. Actions take one of the following two forms.

- Query:  $x\langle\phi\rangle, \overline{x\langle\phi\rangle}$ , where  $x$  : is a pointer name and  $\phi$  is an atomic value.
- Response:  $\langle\phi\rangle, \overline{\langle\phi\rangle}$ , where  $\phi$  is an atomic value.

In each case the atomic value inside the angled brackets is either non-functional (a value passed directly) or functional (a freshly generated pointer name which can be used to pass information back to the generating term). Thus there are eight transitions (input/output, functional/non-functional query/response) generated as follows:

**Query Output.** A program which encounters an output pointer name uses it to send a query containing its argument to the environment, and pushes the



evaluation context onto the execution stack. If the argument to the pointer is non-functional it is passed directly. If it is functional, it is passed as a fresh output pointer.

$$\begin{aligned} E[x v]; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; x; \mathcal{E} &\xrightarrow{\overline{x\langle v \rangle}} \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; x; E[\_]; \mathcal{E}, \\ E[x F]; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; x; \mathcal{E} &\xrightarrow{\overline{x\langle y \rangle}} \mathcal{B}; \mathcal{S}; \mathcal{O}; (y, F); \mathcal{I}; x; E[\_]; \mathcal{E} \end{aligned}$$

**Query Input.** An environment which receives a query  $x\langle\phi\rangle$  evolves into a program by following the input pointer  $x$  and applying the result to  $\phi$ .

$$\begin{aligned} \mathcal{B}; \mathcal{S}; \mathcal{O}; (x, F); \mathcal{I}; \mathcal{E} &\xrightarrow{x\langle u \rangle} F u; \mathcal{B}; \mathcal{S}; \mathcal{O}; (x, F); \mathcal{I}; \mathcal{E}, \\ \mathcal{B}; \mathcal{S}; \mathcal{O}; (x, F); \mathcal{I}; \mathcal{E} &\xrightarrow{x\langle y \rangle} F y; \mathcal{B}; \mathcal{S}; \mathcal{O}; (x, F); \mathcal{I}; y; \mathcal{E} \end{aligned}$$

**Response Output.** A program which produces a value passes this as a response action, either directly (non-functional values) or as a pointer to which it is bound.

$$\begin{aligned} v; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E} &\xrightarrow{\overline{\langle v \rangle}} \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E} \\ F; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E} &\xrightarrow{\overline{\langle y \rangle}} \mathcal{B}; \mathcal{S}; \mathcal{O}; (a, F); \mathcal{I}; \mathcal{E} \end{aligned}$$

**Response Input.** An environment which receives a response evolves into a program by placing it inside an evaluation context popped from the top of the stack.

$$\begin{aligned} \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; E[\_]; \mathcal{E} &\xrightarrow{\langle v \rangle} E[v]; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E} \\ \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; E[\_]; \mathcal{E} &\xrightarrow{\langle y \rangle} E[y]; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; y; \mathcal{E} \end{aligned}$$

We obtain a bipartite LTS with the same set of labels, but having configuration types as nodes, by simply restricting each configuration in the above actions to its type — i.e.

$$\frac{M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E} \xrightarrow{\alpha} \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E}'}{(\text{ty}(M); \text{dom}(\mathcal{O}); \mathcal{I}; \text{ty}(\mathcal{E})) \xrightarrow{\alpha} (\text{dom}(\mathcal{O}'); \mathcal{I}'; \text{ty}(\mathcal{E}'))} \quad \frac{\mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E} \xrightarrow{\alpha} M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E}'}{(\text{dom}(\mathcal{O}); \mathcal{I}; \text{ty}(\mathcal{E})) \xrightarrow{\alpha} (\text{ty}(M); \text{dom}(\mathcal{O}'); \mathcal{I}'; \text{ty}(\mathcal{E}'))}$$

Our full abstraction result is proved for sets of *complete* traces: a trace  $s$  is complete for a program-configuration with stack  $\mathcal{E}$  if the number of response actions in  $s$  is (one) greater than the number of queries plus the depth of the stack  $\mathcal{E}$ . An environment-trace is complete if the number of responses is equal to the number of queries plus the depth of the stack. It is straightforward to show that if  $C \xrightarrow{s} C'$  then  $s$  is complete if and only if  $C'$  is an environment-configuration with empty stack. We write  $\llbracket C \rrbracket$  (resp.  $\llbracket \theta \rrbracket$ ) for the set of complete traces of the configuration  $C$  (resp configuration type  $\theta$ ). By definition, if  $C : \theta$  then  $\llbracket C \rrbracket \subseteq \llbracket \theta \rrbracket$ . Moreover, we will show that if  $s \in \llbracket \theta \rrbracket$  then there exists  $C : \theta$  with  $s \in \llbracket C \rrbracket$ . For a closed term  $M : T$  we define  $\llbracket M \rrbracket = \llbracket M; \_ ; \_ ; \_ ; \_ ; \varepsilon \rrbracket$  — in this case a trace is complete if the number of responses is one greater than the

number of queries. As an example, we give the derivation of a complete trace —  $\overline{\langle y \rangle y \langle g \rangle g \langle \mathbf{tt} \rangle \langle () \rangle \langle () \rangle}$  — of  $\lambda f.(f \mathbf{tt}): (\mathbf{bool} \Rightarrow \mathbf{com}) \Rightarrow \mathbf{com}$ :

$$\begin{aligned} & \lambda f.(f \mathbf{tt}); -; -; -; \varepsilon \xrightarrow{\langle y \rangle} -; -; (y, \lambda f.(f \mathbf{tt})); -; \varepsilon \xrightarrow{y \langle g \rangle} (\lambda f.(f \mathbf{tt})) g; -; -; (y, \lambda f.(f \mathbf{tt})); g; \varepsilon \\ & \longrightarrow h \mathbf{tt}; (h, g); -; (y, \lambda f.(f \mathbf{tt})); g; \varepsilon \longrightarrow (\lambda z.z) (g \mathbf{tt}); (h, g); -; (y, \lambda f.(f \mathbf{tt})); g; \varepsilon \\ & \xrightarrow{g \langle \mathbf{tt} \rangle} (h, g); -; (y, \lambda f.(f \mathbf{tt})); g; \lambda z.z [\_] \xrightarrow{\langle () \rangle} (\lambda z.z) (); (h, g); -; (y, \lambda f.(f \mathbf{tt})); g; \varepsilon \\ & \longrightarrow (); (h, g); -; (y, \lambda f.(f \mathbf{tt})); g; \varepsilon \xrightarrow{\langle () \rangle} (h, g); -; (y, \lambda f.(f \mathbf{tt})); g; \varepsilon. \end{aligned}$$

We will not examine the correspondence between the set of traces of a term and its denotation as a strategy in the games model of [11] in detail. However, readers familiar with game semantics may note that there is a simple correspondence between ( $\alpha$ -equivalence classes of) traces and well-bracketed *alternating justified sequences* in dialogue games: we replace each output/input query/response with a Player/Opponent Question/Answer, and add a “justification pointer” from each query  $x \langle \phi \rangle$  or  $x \langle \bar{\phi} \rangle$  to the query/response in which  $x$  was introduced (if any), and from each response to the pending question. This correspondence sends the trace-set of each term to (the complete sequences of) a deterministic strategy on the associated “arena”. We may show that it preserves the meaning of functional  $\mathcal{L}$ -terms.

### 3.1 Soundness and Completeness

We will now prove inequational soundness for terms in the functional fragment — i.e.  $\llbracket M \rrbracket \subseteq \llbracket N \rrbracket$  implies  $M \lesssim N$ . To show that complete trace inclusion is a precongruence, we prove that the “parallel composition” of two *compatible* functional configurations produces the response  $\langle \bar{() \rangle}$  if and only if evaluating them in combination converges to  $()$ .

**Definition 1.** Let  $P = (M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E})$  be a functional program-configuration, and  $E = (\mathcal{B}'; \mathcal{S}'; \mathcal{O}'; \mathcal{I}'; \mathcal{E}')$  a functional environment-configuration. We say that  $P$  and  $E$  are compatible if:

1.  $\text{dom}(\mathcal{O}) = \mathcal{I}'$  and  $\text{dom}(\mathcal{O}') = \mathcal{I}$ ,
2.  $\text{dom}(\mathcal{B}) \cap \text{dom}(\mathcal{B}') = \text{dom}(\mathcal{S}) \cap \text{dom}(\mathcal{S}') = \emptyset$
3.  $\text{ty}(M) \cdot \text{ty}(\mathcal{E}) = \text{ty}(\mathcal{E}') \cdot \mathbf{com}$  or  $\text{ty}(M) \cdot \text{ty}(\mathcal{E}) \cdot \mathbf{com} = \text{ty}(\mathcal{E}')$ .

if  $P$  and  $E$  are compatible, we write  $P|E \downarrow$  if there exists a trace (a *witness*)  $s$  such that  $s \langle \bar{() \rangle} \in \llbracket P \rrbracket$  and  $s^\perp \in E$  or  $s \in \llbracket P \rrbracket$  and  $s^\perp \langle \bar{() \rangle} \in E$ .

**Definition 2.** The merge of  $P$  and  $E$  is a program-in-environment defined:  $P \bowtie E = (\mathcal{E} \parallel \mathcal{E}') [M]; \mathcal{B}, \mathcal{B}', \mathcal{O}, \mathcal{O}'; \mathcal{S}, \mathcal{S}'$ , where  $(\mathcal{E} \parallel \mathcal{E}') [\_]$  — the unfolding of  $\mathcal{E}, \mathcal{E}'$  — is an evaluation context, defined (up to  $\alpha$ -equivalence) as follows:

- $(\mathcal{E} \parallel \mathcal{E}') [\_] = [\_]$
- $(\mathcal{E} \parallel \mathcal{E}') [E[\_]; \mathcal{E}'] = (\mathcal{E}' \parallel \mathcal{E}) [\lambda x.E[x] \_]$  (where  $x$  is not free in  $E[\_]$ ).

We need to show that if  $P, E$  are compatible configurations then  $P|E \downarrow$  if and only if  $P \bowtie E \Downarrow$ . We prove this by altering the operational semantics

so that it is equivalent with respect to termination, but more closely tracks the interaction between term and environment. Specifically, we replace the rule  $E[x V]; \mathcal{B}, (x, F); \mathcal{S} \longrightarrow E[F V]; \mathcal{B}, (x, F); \mathcal{S}$  with two rules, depending on whether the value  $V$  is functional or non-functional.

$$\begin{aligned} E[x v]; \mathcal{B}, (x, F); \mathcal{S} &\longrightarrow (\lambda z. E[z]) (F v); \mathcal{B}, (x, F); \mathcal{S} \\ E[x G]; \mathcal{B}, (x, F); \mathcal{S} &\longrightarrow (\lambda z. E[z]) (F y); \mathcal{B}, (x, F), (y, G); \mathcal{S} \end{aligned}$$

Writing  $M; \mathcal{B}; \mathcal{S} \Downarrow'$  if  $M; \mathcal{B}; \mathcal{S}$  reduces to  $(); \mathcal{B}'; \mathcal{S}'$  under the modified rules, we may prove equivalence to the original notion of termination using a simulation relation between the two reduction systems.

**Proposition 1.** *For any program-in-environment  $M; \mathcal{B}; \mathcal{S} \Downarrow$  if and only if  $M; \mathcal{B}; \mathcal{S} \Downarrow'$ .*

**Proposition 2.** *If  $P, E$  are compatible configurations then  $P|E \Downarrow$  if and only if  $P \bowtie E \Downarrow'$ .*

*Proof.* We prove the implication from left to right by induction on the length of the witness  $s$  to  $P|E \Downarrow$ . If this is the empty trace then  $\overline{(\ )}$  is a complete trace in  $\llbracket P \rrbracket$  and the empty trace is a complete trace in  $\llbracket E \rrbracket$  — i.e.  $\mathcal{E} = \mathcal{E}' = \varepsilon$  and so  $(\mathcal{E} \parallel \mathcal{E}') [M] = M$  and  $M, \mathcal{B}, \mathcal{S} \rightarrow (), \mathcal{B}'', 'xs''$  and so  $(\mathcal{E} \parallel \mathcal{E}') [M], \mathcal{B} \cup \mathcal{B}', \mathcal{O} \cup \mathcal{O}', \mathcal{S} \cup \mathcal{S}' \Downarrow$  as required. Otherwise  $s = \overline{\alpha}t$  and  $P \rightarrow P' \xrightarrow{\overline{\alpha}} E'$  and  $E \xrightarrow{\alpha} P''$  for some  $P', E'$  such that  $P''|E' \Downarrow$ , and so  $P'' \bowtie E' \Downarrow'$  by hypothesis. By considering each pair of actions  $\overline{\alpha}, \alpha$ , we show that  $P \bowtie E \rightarrow P'' \bowtie E'$  and so  $P \bowtie E \Downarrow'$  as required.

We prove the implication from right to left by induction on the length of the reduction of  $P \bowtie E$  to  $()$ . If  $(\mathcal{E} \parallel \mathcal{E}') [M] = ()$  then the witness to  $P|E \Downarrow$  is  $\varepsilon$ . If  $P \xrightarrow{\overline{\alpha}} P'$  then we may apply the induction hypothesis to  $P', E$ . Otherwise,  $P \xrightarrow{\overline{\alpha}} E'$  and  $E \xrightarrow{\alpha} P'$ . By considering the possible  $\alpha$ , we show that  $P \bowtie E \rightarrow P' \bowtie E'$  and hence  $P''|E' \Downarrow$  by hypothesis, and so  $P|E \Downarrow$  as required.

**Proposition 3 (Soundness).** *If  $\llbracket M \rrbracket \subseteq \llbracket N \rrbracket$  then  $M \lesssim N$ .*

*Proof.* Suppose  $M \not\lesssim N$ . Then by the Context Lemma there exists  $E[\_]; \mathcal{B}; \mathcal{S}$  such that  $E[M]; \mathcal{B}; \mathcal{S} \Downarrow$  and  $E[N]; \mathcal{B}; \mathcal{S} \not\Downarrow$ . Hence there exists  $s\overline{(\ )} \in \llbracket \mathcal{B}; \mathcal{S}; \_ ; \_ ; E[\_] \rrbracket$  such that  $s^\perp \in \llbracket M \rrbracket$  and  $s^\perp \notin N$  — i.e.  $\llbracket M \rrbracket \not\subseteq \llbracket N \rrbracket$  as required.

We now show that our trace semantics is complete for functional terms by proving a *definability* result: every bounded  $\alpha$ -equivalence class of complete traces over a given configuration type which branches only on input actions is the set of traces generated by a configuration of the corresponding type.

**Definition 3.** *A strategy on a configuration type  $\theta$  is a subset  $\sigma \subseteq \llbracket \theta \rrbracket$  which is closed under  $\alpha$ -equivalence  $\square$  and complete prefixes, and which branches only on input actions — i.e. if  $s\overline{\alpha}t, s'\overline{\alpha}'t \in \sigma$  and  $s \sim_\alpha s'$  is even-length then  $sa \sim_\alpha s'a'$ . A strategy on a program-configuration type is a  $p$ -strategy, a strategy on an environment-configuration type is an  $e$ -strategy.*

<sup>1</sup> Traces in  $\llbracket \theta \rrbracket$  are  $\alpha$ -equivalent if they are obtainable by permutation of variables not in  $\theta$ .

It is straightforward to show that if  $C: \theta$  then  $\llbracket C \rrbracket$  is a strategy on  $\theta$ .

**Proposition 4.** *For any configuration type  $\theta$ , every bounded strategy  $\sigma$  on  $\theta$  is the trace of a configuration  $C_\sigma$  of type  $\theta$ .*

*Proof.* By induction on the maximum length of trace, with the additional hypothesis that if  $C_\sigma = (M_\sigma; \mathcal{B}_\sigma; \mathcal{O}_\sigma; \mathcal{I}_\sigma; \mathcal{E}_\sigma)$  or  $(\mathcal{B}_\sigma; \mathcal{O}_\sigma; \mathcal{I}_\sigma; \mathcal{E}_\sigma)$  then  $\mathcal{B} = \emptyset$  and:

- $\mathcal{O}_\sigma = \{(x_i, \lambda z.!q_i z) \mid i \leq m\}$ , where each  $q_i$  is a distinct location name.
- $\mathcal{E}_\sigma = (\lambda z.!r_n z) [\_]; \dots; (\lambda z.!r_1 z) [\_]$ , where each  $r_i$  is a distinct location name.
- $\text{dom}(\mathcal{S}_\sigma) = \{q_i \mid i \leq m\} \cup \{r_j \mid j \leq n\}$

So suppose  $\sigma$  is a  $p$ -strategy, then either  $\sigma = \{\varepsilon\}$  (in which case set  $M = \Omega$ ) or  $\sigma = \{\varepsilon\} \cup \alpha \cdot \tau$ , where  $\tau$  is an  $e$ -strategy, which is definable as above by hypothesis. We define  $\mathcal{O}_\sigma, \mathcal{B}_\sigma, \mathcal{E}_\sigma$  as above, set  $\mathcal{S}_\sigma = \mathcal{S}_\tau | q_1, \dots, q_m, r_1, \dots, r_n$  and define  $M_\sigma$  by analysis of  $\alpha$ :

- If  $\alpha = \overline{y\langle x_{m+1} \rangle}$  then  $M_\sigma = (\lambda z.!r_{n+1} z) (y \lambda z.!q_{m+1} z)$ .
- If  $\alpha = \overline{y\langle v \rangle}$  then  $M_\sigma = (\lambda z.!r_{n+1} z) (y v)$ .
- If  $\alpha = \langle x_{m+1} \rangle$  then  $M_\sigma = \lambda z.!q_{m+1} z$
- If  $\alpha = \langle v \rangle$  then  $M_\sigma = v$ .

Then  $C_\sigma \xrightarrow{\bar{\alpha}} C'$ , for some  $C'$  which is trace-equivalent to  $\tau$ .

An action  $\alpha$  is enabled at  $\theta$  if there exists  $s$  such that  $\alpha s \in \llbracket \theta \rrbracket$ . If  $\sigma$  is an  $e$ -strategy, let  $\hat{\sigma}$  be the function from enabled actions to  $p$ -strategies defined  $\hat{\sigma}(\alpha) = \{s \mid \alpha s \in \sigma\}$ . By inductive hypothesis, for each enabled action  $\alpha$ ,  $\hat{\sigma}(\alpha)$  is definable as a configuration of the specified kind. We define  $\mathcal{O}_\sigma, \mathcal{B}_\sigma, \mathcal{E}_\sigma$  as above, and define  $\mathcal{S}_\sigma$  (with  $\text{dom}(\mathcal{S}_\sigma) = \{q_1, \dots, q_m, r_n\}$ ) as follows: Let  $\text{assign}(\phi, m, n) = q_1 := \mathcal{S}_{\hat{\sigma}(x_i\langle \phi \rangle)}(q_1); \dots; q_m := \mathcal{S}_{\hat{\sigma}(x_i\langle \phi \rangle)}(q_m); r_n := \mathcal{S}_{\hat{\sigma}(x_i\langle \phi \rangle)}(r_n)$ . For each  $x_i : S_i \Rightarrow T_i \in \text{dom}(\mathcal{O})$ ,

- If  $S_i = \text{com}$ , then  $\mathcal{S}_\sigma(q_i) = \lambda z.\text{assign}(\langle \rangle, m, n); M_{\hat{\sigma}(x_i\langle \rangle)}$
- If  $S_i : \text{bool}$  then  $\mathcal{S}_\sigma(q_i) =$   
 $\text{If } z \text{ then assign}(\langle \text{tt} \rangle, m, n); M_{\hat{\sigma}(x_i\langle \text{tt} \rangle)} \text{ else assign}(\langle \text{ff} \rangle, m, n); M_{\hat{\sigma}(x_i\langle \text{ff} \rangle)}$
- $\mathcal{S}_\sigma(q_i) = \lambda y.\text{assign}(y, m, n); M_{\hat{\sigma}(x_i\langle y \rangle)}$  otherwise.

Suppose  $L = S \cdot T \cdot L'$ . Let  $\mathcal{S}_\sigma(r_i) = \lambda x.\Omega$  for  $i < n$  and:

- If  $S : \text{com}$  then  $\mathcal{S}_\sigma(r_n) = \lambda z.\text{assign}(\langle \rangle, m, n); M_{\hat{\sigma}(\langle \rangle)}$ ,
- If  $S : \text{bool}$  then let  $\mathcal{S}_\sigma(r_n) =$   
 $\lambda z.\text{If } z \text{ then assign}(\langle \text{tt} \rangle, m, n-1); M_{\hat{\sigma}(\langle \text{tt} \rangle)} \text{ else assign}(\langle \text{ff} \rangle, m, n-1); M_{\hat{\sigma}(\langle \text{ff} \rangle)}$ ,
- $\mathcal{S}_\sigma(r_n) = \lambda y.\text{assign}(y, m, n-1); M_{\hat{\sigma}(\langle y \rangle)}$  otherwise.

Then for each  $\alpha$  there exists  $C'$  such that  $C_\sigma \xrightarrow{\alpha} C'$ , where  $C'$  is trace-equivalent to  $C_{\hat{\sigma}(\alpha)}$ .

We may now complete the proof of full abstraction.

**Proposition 5.** *For any closed terms  $M, N : T$ , if  $M \lesssim N$  then  $\llbracket M \rrbracket \subseteq \llbracket N \rrbracket$ .*

*Proof.* Suppose  $\llbracket M \rrbracket \not\subseteq \llbracket N \rrbracket$ . Let  $s$  be a minimal length complete sequence in  $\llbracket M \rrbracket$  such that  $s \notin \llbracket N \rrbracket$ . It is straightforward to show that if  $t \in \llbracket \Gamma; \Delta; T; L \rrbracket$  then  $t^\perp \langle \langle \rangle \rangle \in \llbracket \Delta; \Gamma; T : L \cdot \text{com} \rrbracket$ , and so in particular  $s^\perp \langle \langle \rangle \rangle \in \llbracket \_ ; \_ ; T \cdot \text{com} \rrbracket$ . So by Proposition [4](#), there exists a configuration  $E = (\_ ; \mathcal{S}; \_ ; \_ ; E[\_ ]_T)$  such that  $\llbracket E \rrbracket$  consists of the complete prefixes of  $\{s^\perp \langle \langle \rangle \rangle\}$ . Then  $(M; \_ ; \_ ; \_ ; \_ )|E \downarrow$  and  $(N; \_ ; \_ ; \_ ; \_ )|E \not\downarrow$ , and so  $E[M]; \_ ; \mathcal{S} \downarrow$  and  $E[M]; \_ ; \mathcal{S} \not\downarrow$ . Then if the location names occurring in  $E[\_ ]$ ,  $\mathcal{S}$  are  $l_1, \dots, l_k$ , and  $\text{dom}(\mathcal{S}) = (l_{j_1}, \dots, l_{j_n})$ , the required separating context is  $\text{new } l_1 \dots \text{new } l_k.l_{j_1} := \mathcal{S}(l_{j_1}); \dots; l_{j_n} := \mathcal{S}(l_{j_n}); E[\_ ]$ .

## 4 Trace Semantics: Reference Types

The basic LTS fails to capture the behaviour of terms at reference types because it takes no account of implicit interaction between program and environment through shared locations in the store. We extend our LTS so that it is sound and complete for terms of all types by retaining the same basic set of actions but extending them with a component characterizing the action of the term/environment on shared locations. Since programs of purely functional type never can share the name of any location with the environment, the store component for such terms is always empty, so this is (essentially) a conservative extension of the basic LTS as far as such terms are concerned.

We extend the notion of configuration with a set of shared location names  $\mathcal{N}$ , i.e. a program-configuration is a tuple  $(M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E}; \mathcal{N})$  and an environment-configuration a tuple  $(\mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E}; \mathcal{N})$ , with  $\mathcal{N} \subseteq \text{dom}(\mathcal{S})$ <sup>2</sup> in each case. We add the set of shared names to the configuration type — i.e.  $(M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E}; \mathcal{N})$  has type  $(\text{ty}(M); \text{dom}(\mathcal{O}); \mathcal{I}; \text{ty}(\mathcal{E}); \mathcal{N})$ .

Labels of the extended LTS are pairs  $(\alpha, \mathcal{X})$ , where  $\alpha$  is an output/input query/response and  $\mathcal{X}$  is a closed, atom-valued store — a finite partial function from location names to atomic values (basic values, location names, or (fresh) pointer names) such that

- if  $(a, b) \in \mathcal{X}$ , where  $b$  is a location name, then  $b \in \text{dom}(\mathcal{X})$ .
- if  $(a, x), (b, y) \in \mathcal{X}$ , where  $x, y$  are pointer names then  $a \neq b$  implies  $x \neq y$ .

The complement of  $(\alpha, \mathcal{X})$  is  $(\bar{\alpha}, \bar{\mathcal{X}})$ . For output actions  $(\bar{\alpha}, \bar{\mathcal{X}})$  from a program configuration with store  $\mathcal{S}$ , the store component  $\bar{\mathcal{X}}$  is generated by:

- updating the set of shared location names (to include the primary content of the action if it is a location name, and any names now accessible through the store),
- for every shared location  $l$  containing a non-functional value, setting  $\bar{\mathcal{X}}(l) = \mathcal{S}(m)$ .
- for every shared location  $m$  containing a functional value, generating a fresh output pointer  $y$  to  $\mathcal{S}(m)$ , and setting  $\bar{\mathcal{X}}(m) = y$ .

<sup>2</sup> Note that this condition will not always be fulfilled, e.g. if a configuration shares a name for an unassigned location. But it must hold in any configuration reachable from a closed term.

The store component of an input action is used to update the store and set of shared names.

**Definition 4.** *Given a (finite) set of location names  $\mathcal{N}$ , an action  $\alpha$  and a store  $\mathcal{S}$ , we define the set  $\text{acc}(\mathcal{N}, \alpha, \mathcal{S})$  of names in  $\mathcal{S}$  accessible from  $(\mathcal{N}, \alpha)$  to be the least set containing  $\mathcal{N}$  and the content of  $\alpha$  (if it is a location name), and such that for any location names  $(l, m) \in \mathcal{S}$ ,  $l \in \text{acc}(\mathcal{N}, \mathcal{S})$  implies  $m \in \text{acc}(\mathcal{N}, \mathcal{S})$ . Given a set  $S$  of location names, we write  $S_{\text{fn}}$  for the subset of  $S$  consisting of names of locations storing values of functional type and  $S_{\text{nf}}$  for the subset of names of non-functional type,  $\mathcal{S}_{\text{fn}}$  for  $\mathcal{S}[\text{dom}(\mathcal{S})_{\text{fn}}]$  and  $\mathcal{S}_{\text{nf}}$  for  $\mathcal{S}[\text{dom}(\mathcal{S})_{\text{nf}}]$ .*

We may now define the actions of the extended LTS. Unlabelled actions are still the reductions of the operational semantics. For each basic output transition  $M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E} \xrightarrow{\bar{\alpha}} \mathcal{B}; \mathcal{S}; \mathcal{O}'; \mathcal{I}; \mathcal{E}'$  we define an action:

$$M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E}; \mathcal{N} \xrightarrow{(\bar{\alpha}, \mathcal{X})} \mathcal{B}; \mathcal{S}; \mathcal{O}', \{(\mathcal{X}(a), \mathcal{S}(a)) \mid a \in \mathcal{N}'_{\text{fn}}\}; \mathcal{I}; \mathcal{E}'; \mathcal{N}'$$

where  $\mathcal{N}' = \text{acc}(\mathcal{N}, \bar{\alpha}, \mathcal{S})$  and  $\mathcal{X}$  is any atom-valued store such that  $\text{dom}(\mathcal{S}') = \text{dom}(\mathcal{S})$ ,  $\mathcal{X}_{\text{nf}} = \mathcal{S}_{\text{nf}}$ , and  $\text{ran}(\mathcal{X}_{\text{fn}}) \cap (\text{dom}(\mathcal{B}) \cup \text{dom}(\mathcal{O}) \cup \mathcal{I}) = \emptyset$ .

For each basic input transition  $\mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E} \xrightarrow{\alpha} M; \mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}'; \mathcal{E}'$  we define an action:

$$\mathcal{B}; \mathcal{S}; \mathcal{O}; \mathcal{I}; \mathcal{E}; \mathcal{N} \xrightarrow{(\alpha, \mathcal{X})} M; \mathcal{B}; \mathcal{S}[\mathcal{X}]; \mathcal{O}; \mathcal{I}', \text{ran}(\mathcal{X}_{\text{fn}}); \mathcal{E}'; \text{dom}(\mathcal{X})$$

where  $\mathcal{X}$  is any closed atom-valued store such that:  $\mathcal{N} \subseteq \text{dom}(\mathcal{X})$ , if  $\alpha$  contains a location  $l$  then  $l \in \text{dom}(\mathcal{X})$ , and  $\text{ran}(\mathcal{X}_{\text{fn}}) \cap (\text{dom}(\mathcal{B}) \cup \text{dom}(\mathcal{O}) \cup \mathcal{I}') = \emptyset$ .

As in the previous section, the LTS restricts to configuration types, with the same notion of complete trace.

We may establish full abstraction by extending the proofs of soundness and completeness for the functional fragment. We modify the notion of compatibility of configurations  $P = (M; \mathcal{B}; \mathcal{S}; \mathcal{S}; \mathcal{I}; \mathcal{E}; \mathcal{N})$  and  $E = (\mathcal{B}'; \mathcal{S}'; \mathcal{O}'; \mathcal{I}'; \mathcal{E}; \mathcal{N}')$  (Def. 1) by replacing the requirement (2) that  $\mathcal{S}$  and  $\mathcal{S}'$  should be disjoint with  $\text{dom}(\mathcal{S}) \cap \text{dom}(\mathcal{S}') = \mathcal{N}' = \mathcal{N}$ . We redefine the merge operation:  $P \bowtie E = (\mathcal{E} \parallel \mathcal{E}') [M; \mathcal{B}; \mathcal{B}', \mathcal{O}; \mathcal{O}'; \mathcal{S}'[\mathcal{S}]]$ .

To adapt the proof of Proposition 2, we once again modify the operational semantics so that it is equivalent to the original with respect to termination, but more closely reflects interaction in the new LTS. Specifically, we replace the rules for evaluating function application with the following reductions which rebind *all* locations in the store to fresh pointers: — i.e. we replace each rule of the form  $E[FV]; \mathcal{B}; \mathcal{S} \longrightarrow M'; \mathcal{B}'; \mathcal{S}$ , where  $F$  is a pointer name or  $\lambda$ -abstraction, with:

$$E[FV]; \mathcal{B}; \mathcal{S} \longrightarrow M'; \mathcal{B}', \{(\mathcal{S}'(a), \mathcal{S}(a)) \mid a \in \text{dom}(\mathcal{S})_{\text{fn}}\}; \mathcal{S}'$$

where  $\mathcal{S}'$  is any atom-valued store with  $\text{dom}(\mathcal{S}') = \text{dom}(\mathcal{S})$ ,  $\mathcal{S}'_{\text{nf}} = \mathcal{S}_{\text{nf}}$  and  $\text{ran}(\mathcal{S}'_{\text{fn}}) \cap \text{dom}(\mathcal{B}') = \emptyset$ .

**Proposition 6.** *If  $P, E$  are compatible configurations then  $P|E \downarrow$  if and only if  $P \bowtie E \downarrow$ .*

*Proof.* We prove that reduction of  $P \bowtie E$  with respect to the modified semantics tracks the interaction of  $\llbracket P \rrbracket$  and  $\llbracket E \rrbracket$  as in the proof of Proposition 2.

To prove completeness, we extend the proofs of definability for bounded strategies (Proposition 4) to the extended LTS. Again, the proofs are based on those for the functional fragment.

**Proposition 7.** *For any configuration type  $\theta$ , every bounded strategy  $\sigma$  on  $\theta$  is the trace of a configuration  $C_\sigma$  of type  $\theta$ .*

*Proof.* (sketch) Following the proof of Proposition 4,  $p$ -strategies are defined as programs, and  $e$ -strategies as values in the store, whilst output pointers and evaluation contexts on the stack just dereference the associated private locations.

If  $\sigma = \{(\bar{\alpha}, \mathcal{X}) \cdot \tau$  is a  $p$ -strategy, where  $\tau$  is a (definable)  $e$ -strategy then the term  $M_\sigma$  is the assignment of  $\mathcal{X}(a)$  to each location  $a \in \text{dom}(\mathcal{X})_{\text{nf}}$ , followed by the assignment of  $\lambda z.(!q a)$  (where  $q$  is a distinct location  $q$ ) to each location  $a \in \text{dom}(\mathcal{X})_{\text{fn}}$ , followed by the appropriate term producing the action  $\alpha$ .

If  $\sigma$  is an  $e$ -strategy, then for each pointer location  $q$  associated with a given action  $\alpha$ ,  $\mathcal{S}_\sigma(q)$  is a function which dereferences every accessible shared location, and so determines the shared store  $\mathcal{X}^{\mathfrak{S}}$ , and becomes  $M_{\bar{\sigma}(\alpha, \mathcal{X})}$ .

Inequational soundness and completeness follow from Propositions 6 and 7 as in the functional case. Thus we have proved:

**Theorem 1 (Full Abstraction).**  $M \lesssim N$  if and only if  $\llbracket M \rrbracket \subseteq \llbracket N \rrbracket$ .

## References

1. Abramsky, S., Honda, K., McCusker, G.: A fully abstract games semantics for general references. In: LICS 1998. Proceedings of the 13th Annual Symposium on Logic In Computer Science (1998)
2. Berger, M., Honda, K., Yoshida, N.: Sequentiality and the  $\pi$ -calculus. In: Abramsky, S. (ed.) TLCA 2001. LNCS, vol. 2044, Springer, Heidelberg (2001)
3. Berger, M., Honda, K., Yoshida, N.: An observationally complete program logic for imperative higher-order functions. In: Proceedings of LICS 2005, IEEE Computer Society Press, Los Alamitos (2005)
4. Danos, V., Herbelin, H., Regnier, L.: Games semantics and abstract machines. In: LICS 1996. Proceedings of the eleventh International Symposium on Logic In Computer Science (1996)
5. Hyland, J.M.E., Ong, C.-H.L.: Pi-calculus, dialogue games and PCF. In: Proceedings of the 7th ACM Conference on Functional Programming Languages and Computer Architecture, pp. 96–107. ACM Press, New York (1995)
6. Jeffrey, A., Rathke, J.: Towards a theory of bisimulation for local names. In: Proceedings of LICS 1999, IEEE Press, Los Alamitos (1999)

---

<sup>3</sup> Note that because a name may only be (indirectly) shared through location of more complex type, there is still a bound on the number of distinct shared stores which are enabled.

7. Jeffrey, A., Rathke, J.: Java jr.: Fully abstract trace semantics for a core java language. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 423–438. Springer, Heidelberg (2005)
8. Koutavas, V., Wand, M.: Small bisimulations for reasoning about higher-order imperative programs. In: Proceedings of POPL '06, pp. 141–152 (2006)
9. Laird, J.: A game semantics of names and pointers. To appear in *Annals of Pure and Applied Logic* (2006)
10. Sangiorgi, D.: Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms. PhD thesis, University of Edinburgh (1993)
11. Sanjabi, S., Ong, C.-H.L.: Fully abstract semantics of additive aspects by translation. To appear in *proc. AOSD '07* (2007)



# Aliased Register Allocation for Straight-Line Programs Is NP-Complete

Jonathan K. Lee, Jens Palsberg, and Fernando Magno Quintão Pereira

UCLA, University of California, Los Angeles  
{jkenl,palsberg,fernando}@cs.ucla.edu

**Abstract.** Register allocation is NP-complete in general but can be solved in linear time for straight-line programs where each variable has at most one definition point if the bank of registers is homogeneous. In this paper we study registers which may alias: an aliased register can be used both independently or in combination with an adjacent register. Such registers are found in commonly-used architectures such as x86, the HP PA-RISC, the Sun SPARC processor, and MIPS floating point. In 2004, Smith, Ramsey, and Holloway presented the best algorithm for aliased register allocation so far; their algorithm is based on a heuristic for coloring of general graphs. Most architectures with register aliasing allow only *aligned* registers to be combined: for example, the low-address register must have an even number. Open until now is the question of whether working with restricted classes of programs can improve the complexity of aliased register allocation with alignment restrictions. In this paper we show that aliased register allocation with alignment restrictions for straight-line programs is NP-complete.

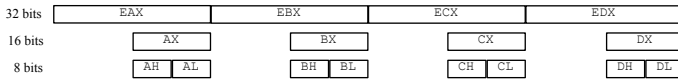
## 1 Introduction

**Register Allocation.** Programmers write most software in high-level programming languages such as C, C++, and Java, and use compilers to translate their programs to a growing variety of hardware, including multicore platforms, graphics processing units, and network processors. To achieve high execution speed, programmers rely on compilers to optimize the program structure and to use registers and memory in clever ways. The latter task, known as *register allocation*, has grown steadily in significance because of the widening gap between the short time to access a register and the longer time to access memory. Today, the register allocator may be among the most important and most complicated parts of a compiler. For example, our experiments with the gcc compiler on the StrongARM architecture shows that a good register allocator typically improves execution speed by a factor of 2.5. A register allocator can also be a significant part of the code of a compiler implementation: 10% for lcc [\[9\]](#) and 12% for gcc 2.95.2.

Most programs use more variables than the number of registers on the target computer. The core of the register allocation problem is to determine whether *all* the program variables can be placed in machine registers. The reason why a register allocator may be able to place a high number of variables in a small number of

registers is that some variables are not live at the same time and so they can share a register. When the need for registers exceeds the number of available registers, the register allocator faces the difficult task of choosing which variables will be placed in registers and which variables will be *spilled*, that is, placed in memory. In this paper we focus on the core register allocation problem and do not discuss spilling of variables.

Chaitin et al. [6] showed that the core register allocation problem is NP-complete by a reduction from the graph coloring problem. The essence of Chaitin et al.'s proof is that every graph is the interference graph of some program. Chaitin et al.'s proof assumes a homogeneous bank of registers, where each register can be used to hold the value of any program variable.



**Aliased Registers.** In this paper we study register allocation for hardware in which the bank of registers is *not* homogeneous. We focus on *aliased registers*: when an assignment to one register name can affect the value of another, such register names *alias* [19]. For example, the figure above shows the set of general purpose registers used in the x86 architecture. The x86 architecture has four general purpose 16-bit registers that can also be used as eight 8-bit registers. Each 8-bit register is called a *low* address or a *high* address. The initial bits of a 16-bit register must be *aligned* with the initial bits of a *low*-address 8-bit register. The x86 architecture allows the combination of two 8-bit registers into one 16 bit register. Another example of aliased registers is the combination of two aligned single precision floating-point registers to form one double-precision register. Examples of architectures with such aliased registers include early versions of HP PA-RISC, the Sun SPARC, and the ARM processors. For a different kind of architecture, Scholz and Eckstein [18] describe experiments with the Carmel 20xxDSP Core, which has six 40 bit accumulators that can also be used as six 32-bit registers or as twelve 16-bit aligned registers.

Architectures that allow unaligned pairing exist but are rare. Some models even allow registers wrapping around, that is, the last and the first registers in the bank combine to form one double register. An example of this type of architecture is the ARM VFP coprocessor.

**Aliased Register Allocation.** We will refer to register allocation for hardware with aliased registers as *aliased register allocation*.

Several research groups have proposed solutions to the aliased register allocation problem. Some of the solutions are based on heuristics for graph coloring [4,5,15,16,17,19], while others are based on integer linear programming [1,10,12,13,14,18] which is flexible enough to describe many architecture irregularities but leads to compile times that are worst-case exponential in the size of the input program.

**Our Results.** We prove that the core aliased register allocation problem with alignment restrictions is NP-complete for straight-line programs where each

variable has at most one definition point. A straight-line program is a sequence of instructions without jumps. Our proof consists of three steps, from 3-SAT via a flow problem and then a coloring problem to our register allocation problem. Our coloring problem *without* alignment restrictions is equivalent to the *shipbuilding* problem; Stockmeyer proved that the shipbuilding problem is NP-complete [11, Application 9.1, p.204]. While we can easily reduce the aligned coloring problem to the unaligned coloring problem (and thereby give an alternative proof of Stockmeyer’s theorem), we have been unsuccessful in doing a reduction in the opposite direction. The aligned case is more restricted than the unaligned case; yet our result shows that the complexity of aliased register allocation in the aligned case is NP-complete.

Our result and Stockmeyer’s result may be surprising because straight-line programs where each variable has at most one definition point are extremely simple. For a homogeneous bank of registers, the core register allocation problem for straight-line programs can be solved in linear time. Our results show that register aliasing is sufficient to bump the complexity to NP-complete.

**Related Work.** At least two other important register allocation problems are NP-complete for straight-line programs: register allocation with precolored registers [3]; and the placement of load and store instructions to transfer values to and from memory [8]. Our proof was inspired in part by a paper of Biró, Hujter, and Tuza [3] who showed how to relate a coloring problem to a flow problem. They used a flow algorithm to solve the precoloring extension problem for interval graphs. Our proof was also inspired by a paper by Even, Itai and Shamir [7] who proved NP-completeness for the multicommodity flow problem.

**Rest of the Paper.** In Section 2 we define our register allocation problem and in Section 3 we define a coloring problem and reduce it to the register allocation problem. In Section 4 we introduce the notion of colored flow for simple graphs, and in Section 5 we reduce the flow problem to the coloring problem. In Section 6 we show how to reduce 3-SAT to the flow problem. The proofs are in the full version of the paper which is available from our website.

## 2 Aliased Register Allocation for Straight-Line Programs

**Programs.** We will define a family of programs that compute with short values and long values. A short value can be represented with half as many bits as a long value. We use  $v$  to range over program variables; a variable is either of type *short* or of type *long*. A variable of type short can only hold short values, and a variable of type long can only hold long values. We define a *statement* by this grammar:

$$\begin{array}{l}
 \text{(Statement) } S ::= \text{short } v = \text{ (definition of } v) \\
 \quad \quad \quad | \text{ long } v = \text{ (definition of } v) \\
 \quad \quad \quad | = v \quad \quad \quad \text{(use of } v)
 \end{array}$$

A statement either defines a variable or uses a variable. We define a *straight-line program* to be a sequence of statements with the property that each variable is

defined only once and used at least once, and every use of a variable comes after its definition.

In program  $S_1; \dots; S_q$ , a variable  $v$  is *live* at statement  $S_j$ , if  $v$  is defined at  $S_i, i \leq j$  and  $v$  is used at  $S_k, j < k$  [2]. Let  $i$  be the index of the statement that defines  $v$ , and let  $k$  be the index of the last statement that uses  $v$ . The *live range* of  $v$  is the half open interval  $[i, k[$ , which includes  $i$  and excludes  $k$ .

If  $v_1, v_2$  are variables and their live ranges have a nonempty intersection, then we say that  $v_1, v_2$  *interfere* [6].

**Aliased Register Allocation.** Suppose we have a machine with  $2K$  registers that each can hold a short value. The registers are called  $r_0, \dots, r_{2K-1}$ ; we call them *short registers*. Suppose further that any two registers  $r_{2i}, r_{2i+1}$ , where  $i \in 0..K-1$ , can be used to hold a long value. Notice the restriction that two registers can hold a long value only if the indices are consecutive and the first index is even; we call this restriction the *alignment restriction*. The alignment restriction models, for example, the rule for how a programmer can use the 8-bit registers on the x86. For example,  $r_4, r_5$  can hold a long value, while  $r_7, r_8$  cannot. We say that the two numbers  $2i, 2i+1$  are *aligned*, and that the two registers  $r_{2i}, r_{2i+1}$  are aligned. We use the notation that for a natural number  $i, \overline{2i} = 2i+1$  and  $\overline{\overline{2i+1}} = 2i$ .

We will study the problem of mapping program variables to machine registers:

CORE ALIASED REGISTER ALLOCATION WITH ALIGNMENT RESTRICTIONS (CARAAR):

**Instance:** a straight line program with  $s$  short variables and  $l$  long variables, and a number  $2K$  of available short registers  $r_0, \dots, r_{2K-1}$ .

**Problem:** Can each short variable be mapped to one of the short registers and can each long variable be mapped to a pair  $r_{2i}, r_{2i+1}, i \in 0..K-1$ , of short registers, such that variables with interfering live ranges are assigned registers that are all different?

### 3 Interval Graphs and Aligned 1-2-Coloring

**Interval Graphs.** We recall the definitions of an *intersection* graph and an *interval* graph [11, p.9].

Let  $\mathcal{S}$  be a family of nonempty sets. The intersection graph of  $\mathcal{S}$  is obtained by representing each set in  $\mathcal{S}$  by a vertex and connecting two vertices by an edge if and only if their corresponding sets intersect. An *interval* graph is an intersection graph of a family of subintervals of an interval of the real numbers. We will examine interval graphs with two kinds of intervals, called short and long intervals; we call such graphs *aliased interval graphs*.

**Aligned 1-2 Coloring.** We will study a variant of graph coloring which we call *aligned 1-2-coloring*. We will use natural numbers as colors; for example, if we have  $2K$  colors, then we will use  $0, 1, \dots, 2K-1$  as the colors. We will color each vertex, that is, each interval. We will use the terms “short interval” and “short vertex” interchangeably; and similarly for “long interval” and “long vertex”. We define a *1-2-coloring* to be a mapping that assigns one color to each short vertex and two

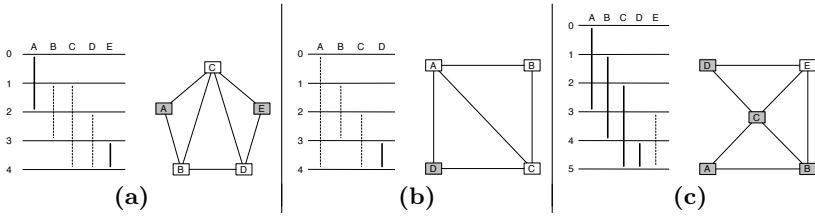


Fig. 1. Aliased interval graphs

colors  $i, i + 1, i \in 0..2K - 2$ , to each long vertex such that adjacent vertices have colors that are all different. We define an *aligned 1-2-coloring* to be a 1-2-coloring that assigns two aligned colors to each long vertex.

ALIGNED 1-2-COLORING OF ALIASED INTERVAL GRAPHS (A12CAIG):

**Instance:** an aliased interval graph and a number  $2K$  of colors.

**Problem:** Find an aligned 1-2-coloring that uses  $2K$  colors.

We will show that A12CAIG is NP-complete.

**From aligned 1-2 coloring to aliased register allocation.** We now present a reduction of aligned 1-2-coloring of aliased interval graphs to aliased register allocation with alignment restrictions. The key step is to show that any aliased interval graph is the interference graph of one of our straight-line programs. We first define a convenient representation of interval graphs. We say that an interval graph is *program like* if (1) all the intervals are of the form  $[u, v[$ , (2) the start and end points of the intervals form the set  $1..2q$ , where  $q$  is the number of intervals, (3) no two intervals start at the same point, (4) no two intervals end at the same point, and (5) no interval starts in the point where another interval ends.

**Proposition 1.** *A graph is an interval graph if and only if it is a program-like interval graph.*

From a program-like interval graph  $H$ , we construct a program  $P = S_1; \dots; S_{2q}$  as follows. Define

$$\forall i \in 1..2q : S_i = \begin{cases} \text{short } v_I = & \text{if the short interval } I \text{ begins at } i \\ \text{long } v_I = & \text{if the long interval } I \text{ begins at } i \\ = v_I & \text{if the interval } I \text{ ends at } i \end{cases}$$

**Lemma 1.** *H is the interference graph of P.*

**Example.** Let us explain why aligned 1-2-coloring is a nontrivial problem. Figure 1 shows three aliased interval graphs; each graph is displayed both as a collection of intervals and in a conventional way. In the left part of each figure, we use fat lines to denote long intervals and we use dashed lines to denote short intervals. In the right part of each figure, we use shaded boxes to denote “long” vertices (representing long intervals) and we use white boxes to denote “short” vertices (representing short intervals).

A standard interval graph has the property that the size of the largest clique is equal to the minimal number of colors [11, p.17]. Aligned coloring of an aliased interval graph does not necessarily have that property. For example, Figure 1(a) shows a graph for which the minimal 1-2-coloring uses four colors:  $A = \{0, 1\}, B = 2, C = 3, D = 0, E = \{1, 2\}$ , while the minimal aligned 1-2-coloring uses five colors:  $A = \{0, 1\}, B = 2, C = 3, D = 4, E = \{0, 1\}$ . Notice that the largest clique is of size 3; even if we treat long variables as counting as two nodes, the largest clique is of size 4.

A standard interval graph has the property that we can optimally color the graph by applying greedy coloring to any perfect elimination ordering of the vertices. (In a perfect elimination ordering, the neighbors of a node  $v$  that come before  $v$  in the ordering form a clique [11, p.82].) An aliased interval graph does not necessarily have that property. For example, Figure 1(b) shows a graph for which we have the perfect elimination ordering  $\langle A, B, C, D \rangle$  that leads greedy coloring to produce an aligned 1-2-coloring with five colors:  $A = 0, B = 1, C = 2, D = \{4, 5\}$ . If we drop the alignment restriction, greedy coloring again produces a 1-2-coloring with five colors:  $A = 0, B = 1, C = 2, D = \{3, 4\}$ . However, in both the aligned and unaligned cases, there exists an optimal assignment using just four colors:  $A = 0, B = 2, C = 1, D = \{2, 3\}$ .

We might try an algorithm that first applies greedy coloring to the short intervals and then proceeds to color the longs. That does not necessarily lead to an optimal 1-2-coloring. For example, Figure 1(b) shows a graph for which we have already studied the perfect elimination ordering  $\langle A, B, C, D \rangle$  in which all the short intervals come before the long intervals. So, we will get the same suboptimal colorings as above.

Alternatively, we might try to first apply greedy coloring to the long intervals, and then proceed to color the shorts. That method is not optimal either. For example, Figure 1(c) shows a graph for which the “longs-first” method produces the 1-2-coloring  $A = \{0, 1\}, B = \{2, 3\}, C = \{4, 5\}, D = \{0, 1\}, E = 6$ . Notice that the 1-2-coloring is also an aligned 1-2-coloring. However, in both the aligned and unaligned cases, an optimal assignment uses just six colors:  $A = \{0, 1\}, B = \{2, 3\}, C = \{4, 5\}, D = \{2, 3\}, E = 0$ .

None of the simple methods work because the aligned and unaligned 1-2-coloring problems are NP-complete.

## 4 Simple Graphs, Straight Cuts, and Colored Flows

Let  $(V, E, Source, Sink, c)$  be a directed graph with vertex set  $V$ , edge set  $E$ , distinguished vertices  $Source, Sink \in V$ , and a capacity function  $c : E \rightarrow Nat$ , where  $Nat$  denotes the natural numbers.

A flow is a function  $f : E \rightarrow Nat$ , such that

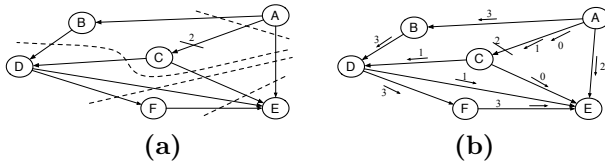
$$\forall (u, v) \in E : f(u, v) \leq c(u, v) \quad (\text{Capacity})$$

$$\forall v \in V \setminus \{Source, Sink\} : \sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w) \quad (\text{Conservation})$$

The value of a flow is the sum of the flows of the edges that reach *Sink*. A *maximal flow* is flow  $f$  such that for any flow  $g$ , the value of  $g$  is less than or equal to the value of  $f$ .

We define a set of vertices  $S$  to be *backwards closed* if  $\forall v \in S : \text{if } (u, v) \in E, \text{ then } u \in S$ . We also define a set of vertices  $T$  to be *forwards closed* if  $\forall u \in T : \text{if } (u, v) \in E, \text{ then } v \in T$ . A cut  $(S, T)$  is a partition of  $V$  such that  $Source \in S, Sink \in T, S \cap T = \emptyset, \text{ and } S \cup T = V$ . The capacity of a cut  $(S, T)$ , written  $c(S, T)$  is given by the formula:  $c(S, T) = \sum_{(u,v) \in E, u \in S, v \in T} c(u, v)$ , which says that the capacity of the cut is the sum of the capacities of the edges that cross the cut from  $S$  to  $T$ . A *straight cut* is a cut  $(S, T)$  such that  $S$  is backwards closed and  $T$  is forwards closed. We define a *simple graph* to be an acyclic graph  $(V, E, Source, Sink, c)$  in which  $Source$  has no incoming edges,  $Sink$  has no outgoing edges, and where all the straight cuts have the same capacity.

Part (a) of the figure below shows a simple graph. Each dashed line marks a straight cut. Each edge with nonunit capacity is marked with a small bar and its capacity; unlabeled edges have unit capacity.



**Lemma 2.** All straight cuts have the same capacity if and only if  $\forall v \in V \setminus \{Source, Sink\} : \sum_{(u,v) \in E} c(u, v) = \sum_{(v,w) \in E} c(v, w)$ .

**Lemma 3.** In a simple graph,  $c$  is the maximal flow.

We say that an element of  $0..K - 1$  is a *color*. We define a *colored flow* for a simple graph with every straight cut of capacity  $K$  as a function  $h : E \rightarrow 2^{0..K-1}$  such that  $\lambda e. |h(e)|$  is a flow and for any straight cut  $(S, T)$ , we have  $\cup_{(u,v) \in E, u \in S, v \in T} h(u, v) = 0..K - 1$ . Thus, for any straight cut, every color is used exactly once in the coloring of the edges that cross the cut. Notice that every color is used *at most* once because the straight cut has capacity  $K$ . We use Lemma 3 to justify the terminology that a *maximal colored flow* is a colored flow with the property that  $\lambda e. |h(e)| = c$ .

Part (b) of the figure above shows an example of colored flow.

**Lemma 4.** For a simple graph,  $h$  is a colored flow if and only if  $\lambda e. |h(e)|$  is a flow,  $\forall v \in V \setminus \{Source, Sink\} : \cup_{(u,v) \in E} h(u, v) = \cup_{(v,w) \in E} h(v, w)$ , and  $\exists$  straight cut  $(S, T)$  such that  $\cup_{(u,v) \in E, u \in S, v \in T} h(u, v) = 0..K - 1$ .

**Aligned colored flow.** Suppose we have a simple graph  $(V, E, Source, Sink, c)$  with all straight cuts of capacity  $2K$ , a function  $A : E \rightarrow Boolean$ , and a number  $2K$  of colors  $0, \dots, 2K - 1$ . We define an *aligned colored flow* to be a colored flow  $h$  such that if  $A(e) = true$  and  $2 \leq c(e)$ , then  $\exists i : 0 \leq i \leq K - 1 \wedge \{2i, 2i + 1\} \subseteq h(e)$ . Intuitively, the function  $A$  indicates that an edge  $e$  with a capacity of at least two requires  $h$  to assign  $e$  the colors  $2i$  and  $2i + 1$ , among others.

MAXIMAL, ALIGNED COLORED FLOW:

**Instance:**  $(G, 2K, A)$ , where  $G$  is a simple graph  $(V, E, Source, Sink, c)$  with all straight cuts of capacity  $2K$ , and  $A : E \rightarrow Boolean$ .

**Problem:** Find a maximal, aligned colored flow that uses  $2K$  colors.

## 5 From Maximal, Aligned Colored Flow to Aligned 1-2 Coloring

In this section we present a reduction of the maximal, aligned colored flow problem to aligned 1-2-coloring of aliased interval graphs. Let  $(G, 2K, A)$  be an instance of the maximal, aligned colored flow problem, where  $G = (V, E, Source, Sink, c)$ . From  $(G, 2K, A)$  we construct an aliased interval graph  $H$  in the following way.

First we sort  $V$  into a topological order with *Source* first and *Sink* last. We can do that because  $G$  is a simple graph so *Source* has no incoming edges and *Sink* has no outgoing edges.

Next we define an injective function  $\ell : V \rightarrow Nat$  such that if  $v_1$  is less than  $v_2$  in the topological ordering, then  $\ell(v_1) < \ell(v_2)$ . The numbers assigned by  $\ell$  to the vertices of  $G$  will be the start and end points of the intervals in  $H$ . The intervals of  $H$  are defined as follows. For each  $(u, v) \in E$  such that  $A(u, v) = true$ , we create one long interval  $[\ell(u), \ell(v)[$  and  $c(u, v) - 2$  short intervals  $[\ell(u), \ell(v)[$ . For each  $(u, v) \in E$  such that  $A(u, v) = false$ , we create  $c(u, v)$  short intervals  $[\ell(u), \ell(v)[$ .

**Lemma 5.**  $(G, 2K, A)$  has a maximal, aligned colored flow if and only if  $H$  has an aligned 1-2-coloring.

## 6 From 3-SAT to Maximal, Aligned Colored Flow

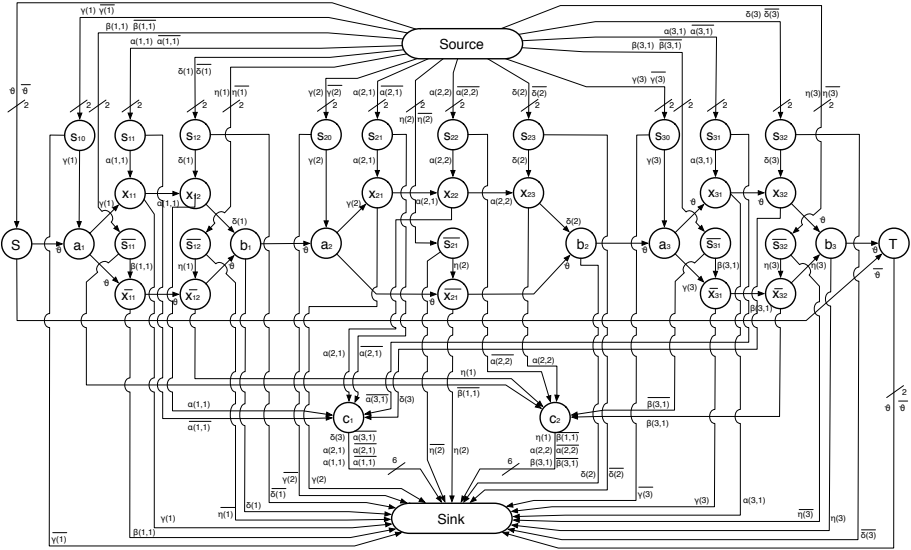
In this section we present a reduction of 3-SAT to the maximal, aligned colored flow problem. Let  $F = \bigwedge_{j=1}^m c_j, c_j = l_{j1} \vee l_{j2} \vee l_{j3}$  be a formula with  $n$  Boolean variables  $x_1, \dots, x_n$  and  $m$  clauses  $c_1, \dots, c_m$ ; each literal,  $l_{j1}$  or  $l_{j2}$  or  $l_{j3}$ , is either a variable or the negation of a variable, and in each clause the three literals are distinct. Let  $p_i$  be the number of occurrences of  $x_i$ , and let  $q_i$  be the number of occurrences of  $\bar{x}_i$ . We index a certain set of vertices using  $i$  and  $k$ , where  $1 \leq i \leq n$  and  $1 \leq k \leq p_i + 1$ . We index another set of vertices using  $i$  and  $h$ , where  $1 \leq i \leq n$  and  $1 \leq h \leq q_i + 1$ . For convenience, we define  $p_0 = 0$  and  $q_0 = 0$ .

From  $F$  we construct a simple graph  $G = (V, E, Source, Sink, c)$ . The graph is akin to the graph used by Even, Itai and Shamir [7, Section 4] in their proof of NP-completeness for the multicommodity flow problem. Figure 2 shows a listing of the

| Vertex        | Cap | Vertex | Cap | Vertex   | Cap | Vertex   | Cap | Vertex         | Cap | Vertex | Cap |
|---------------|-----|--------|-----|----------|-----|----------|-----|----------------|-----|--------|-----|
| <i>Source</i> | 0   | $S$    | 2   | $c_j$    | 6   | $x_{ik}$ | 2   | $\bar{x}_{ih}$ | 2   | $a_i$  | 2   |
| <i>Sink</i>   | 2K  | $T$    | 2   | $s_{i0}$ | 2   | $s_{ik}$ | 2   | $\bar{s}_{ih}$ | 2   | $b_i$  | 2   |

Fig. 2. Vertices with incoming capacities;  $K = 3m + 3n + 1$





**Fig. 3.** Construction of a simple graph from  $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$ . Edges with a labeled capacity require aligned colors. The edge labels are explained in Figure 5 (a).

vertices in  $V$ , along with, for each vertex, the sum of the capacities of the in-going edges (which, because the graph is simple, is equal to the sum of the capacities of the outgoing edges, for all vertices except *Source*, *Sink*). Figure 4 shows a listing of the edges in  $E$ . Figure 4 also shows a set of colors for each edge; we will need that later. Figure 3 illustrates the graph constructed from the formula  $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$ . Let us now briefly walk through the construction of the graph. We denote an edge from  $u$  to  $v$  with a capacity  $c$  as  $(u \xrightarrow{c} v)$

We have the vertices *Source*, *Sink*, two vertices  $S$  and  $T$ , one vertex  $c_j$  for each clause, and vertices  $s_{i0}, x_{ik}, s_{ik}, \bar{x}_{ih}, \bar{s}_{ih}, a_i, b_i$ , for each variable  $x_i$ . For each variable  $x_i$ , we construct a lobe. We add the edges  $(x_{iu} \xrightarrow{1} x_{i(u+1)})$ ,  $1 \leq u \leq p_i$  to form an upper path. We construct a lower path with the edges  $(\bar{x}_{iv} \xrightarrow{1} \bar{x}_{i(v+1)})$ ,  $1 \leq v \leq q_i$ . We connect these paths to  $a_i$  and  $b_i$  to form the lobe by adding the edges  $(a_i \xrightarrow{1} x_{i1}), (a_i \xrightarrow{1} \bar{x}_{i1}), (x_{i(p_i+1)} \xrightarrow{1} b_i)$ , and  $(\bar{x}_{i(q_i+1)} \xrightarrow{1} b_i)$ .

Next we are going to make several edges that have alignment requirements. For each  $s_{ik}$ , we create an edge  $(Source \xrightarrow{2} s_{ik})$  with a capacity two. Likewise for all the  $s_{i0}$  and the  $\bar{s}_{ih}$  vertices. Next we will add an edge  $(Source \xrightarrow{2} S)$  also with capacity of two. In total we have made a capacity of  $2(3m + 3n + 1)$  leaving the source. We want to make  $G$  simple, so there must be capacities of two leaving each of these vertices and eventually reaching the *Sink*. We will create some more aligned edges which will now connect certain vertices to *Sink*. For each of the  $c_j$  vertices, we create the edges  $(c_j \xrightarrow{6} Sink)$  with a capacity of six and finally we add  $(T \xrightarrow{2} Sink)$  with a capacity of two. Now all that remains to make the graph simple is to connect the  $S, s_{ik}$ , and  $\bar{s}_{ih}$  vertices to  $T, c_j$ , and *Sink*.

| Edge  | Color                                   | Edge   | Color  |
|---|---|--|--|
| $x_{iu} \xrightarrow{1} x_{i(u+1)}$             | $\psi(x_i) ? \alpha(i, u) : \theta$     | $s_{i0} \xrightarrow{1} Sink$  | $\overline{\gamma(i)}$                           |
| $\bar{x}_{iv} \xrightarrow{1} \bar{x}_{i(v+1)}$ | $\psi(x_i) ? \theta : \beta(i, v)$      | $s_{i0} \xrightarrow{1} a_i$   | $\gamma(i)$                                      |
| $a_i \xrightarrow{1} x_{i1}$                    | $\psi(x_i) ? \gamma(i) : \theta$        | $b_i \xrightarrow{1} Sink$   | $\psi(x_i) ? \delta(i) : \eta(i)$                |
| $a_i \xrightarrow{1} \bar{x}_{i1}$              | $\psi(x_i) ? \theta : \gamma(i)$        | $s_{iu} \xrightarrow{1} x_{iu}$  | $\alpha(i, u)$                                   |
| $x_{i(p_i+1)} \xrightarrow{1} b_i$              | $\psi(x_i) ? \delta(i) : \theta$        | $s_{i(p_i+1)} \xrightarrow{1} x_{i(p_i+1)}$  | $\delta(i)$                                      |
| $\bar{x}_{i(q_i+1)} \xrightarrow{1} b_i$        | $\psi(x_i) ? \theta : \eta(i)$          | $\bar{s}_{iv} \xrightarrow{1} \bar{x}_{iv}$  | $\beta(i, u)$                                    |
| $Source \xrightarrow{2} s_{iu}$                 | $\alpha(i, u), \overline{\alpha(i, u)}$ | $\bar{s}_{i(q_i+1)} \xrightarrow{1} \bar{x}_{i(q_i+1)}$                                    | $\eta(i)$  |
| $Source \xrightarrow{2} s_{i(p_i+1)}$           | $\delta(i), \overline{\delta(i)}$       | $s_{i(p_i+1)} \xrightarrow{1} Sink$  | $\delta(i)$                                      |
| $Source \xrightarrow{2} \bar{s}_{iv}$           | $\beta(i, u), \overline{\beta(i)}$      | $\bar{s}_{i(q_i+1)} \xrightarrow{1} Sink$  | $\eta(i)$  |
| $Source \xrightarrow{2} \bar{s}_{i(q_i+1)}$     | $\eta(i), \overline{\eta(i)}$           | $s_{iu} \xrightarrow{1} c_j$<br>if $\text{occ}(u, x_i, c_j)$                               | $\alpha(i, u)$                                   |
| $Source \xrightarrow{2} s_{i0}$                 | $\gamma(i), \overline{\gamma(i)}$       | $\bar{s}_{iv} \xrightarrow{1} c_j$<br>if $\text{occ}(v, \bar{x}_i, c_j)$                   | $\beta(i, u)$                                    |
| $Source \xrightarrow{2} S$                      | $\theta, \overline{\theta}$             | $x_{i(u+1)} \xrightarrow{1} c_j$<br>if $\text{occ}(u, x_i, c_j), (u \neq p_i)$             | $\psi(x_i) ?$<br>$\alpha(i, u) : \alpha(i, u+1)$ |
| $c_j \xrightarrow{6} Sink$                      | See Figure 5(b)                         | $x_{i(p_i+1)} \xrightarrow{1} c_j$<br>if $\text{occ}(p_i, x_i, c_j)$                       | $\psi(x_i) ?$<br>$\alpha(i, p_i) : \delta(i)$    |
| $T \xrightarrow{2} Sink$                        | $\theta, \overline{\theta}$             | $\bar{x}_{i(v+1)} \xrightarrow{1} c_j$<br>if $\text{occ}(v, \bar{x}_i, c_j), (v \neq q_i)$ | $\psi(x_i) ?$<br>$\beta(i, v) : \beta(i, v+1)$   |
| $S \xrightarrow{1} T$                           | $\overline{\theta}$                     | $\bar{x}_{i(q_i+1)} \xrightarrow{1} c_j$<br>if $\text{occ}(q_i, \bar{x}_i, c_j)$           | $\psi(x_i) ? \eta(i) : \beta(i, q_i)$            |
| $b_l \xrightarrow{1} a_{l+1}$                   | $\theta$                                | $x_{i1} \xrightarrow{1} Sink$  | $\psi(x_i) ? \gamma(i) : \alpha(i, 1)$           |
| $S \xrightarrow{1} a_1$                         | $\theta$                                | $\bar{x}_{i1} \xrightarrow{1} Sink$  | $\psi(x_i) ? \gamma(i) : \beta(i, 1)$            |
| $b_n \xrightarrow{1} T$                         | $\theta$                                |  |  |

**Fig. 4.** Edge construction;  $\psi(x_i)d_T : d_F$  denotes that if  $\psi(x_i) = true$  then the assigned color is  $d_T$  and if  $\psi(x_i) = false$  then the assigned color is  $d_F$ .  $1 \leq i \leq n, 1 \leq j \leq m, 1 \leq u \leq p_i, 1 \leq v \leq q_i, 1 \leq l \leq n-1$ . An expression  $\text{occ}(u, x_i, c_j)$  means that the  $u^{\text{th}}$  occurrence of  $x_i$  appears in  $c_j$ .

We will first add edges to send the current excess capacity at  $S$  to  $T$ . We will add a direct edge ( $S \xrightarrow{1} T$ ) to get one unit to  $T$ . To get the other unit to  $T$ , we will connect the lobes serially, by adding the edges  $(b_l, a_{l+1}), 1 \leq l \leq n-1$ . Finally, we add  $(S \xrightarrow{1} a_1)$  and  $(b_n \xrightarrow{1} T)$ , resulting in a path to send the other unit of capacity to  $T$  and two units of capacity reaching  $Sink$ .

The  $a_i$  and  $b_i$  vertices still have an imbalance of capacity and must have edges to supply capacity or drain it. To correct for these imbalances, we add the edges  $(s_{i0} \xrightarrow{1} Sink), (s_{i0} \xrightarrow{1} a_i)$ , and  $(b_i \xrightarrow{1} Sink)$ . This results in a current total of  $2n+2$  units of capacity reaching  $Sink$ , and the vertices on the lobe balanced.

We will now connect the remaining  $s_{ik}$  and  $\bar{s}_{ih}$  vertices to the  $c_j$  vertices and  $Sink$ . We add the edges  $(s_{ik} \xrightarrow{1} x_{ik})$  and  $(\bar{s}_{ih} \xrightarrow{1} \bar{x}_{ih})$  which will send one unit of capacity from each of these vertices to the corresponding vertices on the lobe. The other units from the  $s_{ik}$  and  $\bar{s}_{ih}$  vertices will be sent to either some  $c_j$  vertex or directly to  $Sink$ . We add the edges  $(s_{i(p_i+1)} \xrightarrow{1} Sink)$  and  $(\bar{s}_{i(q_i+1)} \xrightarrow{1} Sink)$ , which now results in an additional  $2n$  units of capacity reaching  $Sink$  for a running total

| $\alpha(i, k) = 2(\sum_{z=1}^i p_{z-1} + k - 1)$ $\beta(i, h) = 2(\sum_{z=1}^n p_z + \sum_{z=1}^i q_{z-1} + h - 1)$ $\gamma(i) = 2(3m + i - 1)$ $\delta(i) = 2(3m + n + i - 1)$ $\eta(i) = 2(3m + 2n + i - 1)$ $\theta = 2(3m + 3n)$ | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="padding: 5px;">Condition</th> <th style="padding: 5px;">Colors</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;"><math>x_i</math> is <math>u^{th}</math> occ., <math>\psi(x_i) = T</math></td> <td style="padding: 5px;"><math>\alpha(i, u), \alpha(i, u)</math></td> </tr> <tr> <td style="padding: 5px;"><math>x_i</math> is <math>u^{th}</math> occ., <math>\psi(x_i) = F</math>,<br/><math>u \neq p_i</math></td> <td style="padding: 5px;"><math>\alpha(i, u + 1),</math><br/><math>\alpha(i, u)</math></td> </tr> <tr> <td style="padding: 5px;"><math>x_i</math> is <math>p_i^{th}</math> occ., <math>\psi(x_i) = F</math></td> <td style="padding: 5px;"><math>\delta(i), \alpha(i, p_i)</math></td> </tr> <tr> <td style="padding: 5px;"><math>\bar{x}_i</math> is <math>v^{th}</math> occ., <math>\psi(x_i) = F</math></td> <td style="padding: 5px;"><math>\beta(i, v), \beta(i, v)</math></td> </tr> <tr> <td style="padding: 5px;"><math>\bar{x}_i</math> is <math>v^{th}</math> occ., <math>\psi(x_i) = T</math>,<br/><math>v \neq q_i</math></td> <td style="padding: 5px;"><math>\beta(i, v + 1),</math><br/><math>\beta(i, v)</math></td> </tr> <tr> <td style="padding: 5px;"><math>\bar{x}_i</math> is <math>q_i^{th}</math> occ., <math>\psi(x_i) = T</math></td> <td style="padding: 5px;"><math>\eta(i), \beta(i, q_i)</math></td> </tr> </tbody> </table> | Condition | Colors | $x_i$ is $u^{th}$ occ., $\psi(x_i) = T$ | $\alpha(i, u), \alpha(i, u)$ | $x_i$ is $u^{th}$ occ., $\psi(x_i) = F$ ,<br>$u \neq p_i$ | $\alpha(i, u + 1),$<br>$\alpha(i, u)$ | $x_i$ is $p_i^{th}$ occ., $\psi(x_i) = F$ | $\delta(i), \alpha(i, p_i)$ | $\bar{x}_i$ is $v^{th}$ occ., $\psi(x_i) = F$ | $\beta(i, v), \beta(i, v)$ | $\bar{x}_i$ is $v^{th}$ occ., $\psi(x_i) = T$ ,<br>$v \neq q_i$ | $\beta(i, v + 1),$<br>$\beta(i, v)$ | $\bar{x}_i$ is $q_i^{th}$ occ., $\psi(x_i) = T$ | $\eta(i), \beta(i, q_i)$ |
|--|--|-----------|--------|---|------------------------------|---|---------------------------------------|---|-----------------------------|---|----------------------------|---|-------------------------------------|---|--------------------------|
| Condition  | Colors   |           |        |   |                              |   |                                       |   |                             |   |                            |   |                                     |   |                          |
| $x_i$ is $u^{th}$ occ., $\psi(x_i) = T$  | $\alpha(i, u), \alpha(i, u)$   |           |        |   |                              |   |                                       |   |                             |   |                            |   |                                     |   |                          |
| $x_i$ is $u^{th}$ occ., $\psi(x_i) = F$ ,<br>$u \neq p_i$  | $\alpha(i, u + 1),$<br>$\alpha(i, u)$  |           |        |   |                              |   |                                       |   |                             |   |                            |   |                                     |   |                          |
| $x_i$ is $p_i^{th}$ occ., $\psi(x_i) = F$  | $\delta(i), \alpha(i, p_i)$  |           |        |   |                              |   |                                       |   |                             |   |                            |   |                                     |   |                          |
| $\bar{x}_i$ is $v^{th}$ occ., $\psi(x_i) = F$  | $\beta(i, v), \beta(i, v)$   |           |        |   |                              |   |                                       |   |                             |   |                            |   |                                     |   |                          |
| $\bar{x}_i$ is $v^{th}$ occ., $\psi(x_i) = T$ ,<br>$v \neq q_i$  | $\beta(i, v + 1),$<br>$\beta(i, v)$  |           |        |   |                              |   |                                       |   |                             |   |                            |   |                                     |   |                          |
| $\bar{x}_i$ is $q_i^{th}$ occ., $\psi(x_i) = T$  | $\eta(i), \beta(i, q_i)$   |           |        |   |                              |   |                                       |   |                             |   |                            |   |                                     |   |                          |

**Fig. 5.** (a) Abbreviations. (b) For each literal in  $c_j$ , the set of colors in  $h(c_j \xrightarrow{e} Sink)$ .

of  $4n + 2$ . The remaining vertices add an edge ( $s_{iu} \xrightarrow{1} c_j$ ) if the  $u^{th}$  appearance of  $x_i$  occurs in  $c_j$ . For the  $\bar{s}_{iv}$  vertices, we add similar edges. From these edges we get  $3m$  units of capacity reaching *Sink*, because each of these edges corresponds to a clause, and each clause has exactly three literals in it. All that remains is to drain the single unit of capacity currently residing at the  $x_{ik}$  and  $\bar{x}_{ih}$  vertices and we will have our simple graph. We add the edges ( $x_{i(u+1)} \xrightarrow{1} c_j$ ) if the  $u^{th}$  appearance of  $x_i$  occurs in  $c_j$  as well as ( $x_{i(v+1)} \xrightarrow{1} c_j$ ) if the  $v^{th}$  appearance of  $\bar{x}_i$  occurs in  $c_j$ . This results in another  $3m$  units of capacity reaching *Sink*. Finally, the last  $2n$  units will be supplied by the edges ( $x_{i1} \xrightarrow{1} Sink$ ) and ( $\bar{x}_{i1} \xrightarrow{1} Sink$ ).

**Lemma 6.** *G is simple.*

Let  $A : E \rightarrow Boolean$  be given by mapping each edge  $e$  to the Boolean value obtained by computing  $2 \leq c(e)$ , where  $c$  is the capacity function specified implicitly in Figure 4.

**Lemma 7.** *F is satisfiable if and only if  $(G, 2(3m + 3n + 1), A)$  has a maximal, aligned colored flow.*

## 7 Main Result and Conclusion

**Theorem 1.** *For straight-line programs, the core aliased register allocation problem with alignment restrictions is NP-complete.*

*Proof.* Firstly, we see the problem is in NP because a register assignment can be verified in polynomial time. We have a chain of reductions from 3-SAT to maximal, aligned colored flow (Lemma 7), from maximal, aligned colored flow to aligned 1-2 coloring (Lemma 5), and from aligned 1-2 coloring to aliased register allocation (Lemma 1). □

We have shown that aliased register allocation with alignment restrictions is difficult, even for straight-line programs where each variable has at most one definition point. Our result confirms the need for the heuristics and worst-case exponential time methods that are used today.

In this paper we have considered register allocation as a decision problem. We can also view register allocation as an optimization problem: minimize the number of registers. Open problem: give nontrivial upper and lower bounds on the approximability of our register allocation problem. For example, is our register allocation problem APX-hard?

## References

1. Appel, A.W., George, L.: Optimal spilling for CISC machines with few registers. In: PLDI, pp. 243–253. ACM Press, New York (2001)
2. Appel, A.W., Palsberg, J.: Modern Compiler Implementation in Java. Cambridge University Press, Cambridge (2002)
3. Biró, M., Hujter, M., Tuza, Z.: Precoloring extension. I:  $\hat{E}$ interval graphs. In: Discrete Mathematics, p. 267. ACM Press, New York (1992) Special volume (part 1) to mark the centennial of Julius Petersen's Die theorie der regularen graphs
4. Briggs, P.: Register Allocation via Graph Coloring. PhD thesis, Rice University (1992)
5. Briggs, P., Cooper, K., Torczon, L.: Coloring register pairs. ACM Letters on Programming Languages 1(1), 3–13 (1992)
6. Chaitin, G.J., Auslander, M.A., Chandra, A.K., Cocke, J., Hopkins, M.E., Markstein, P.W.: Register allocation via coloring. Computer Languages 6, 47–57 (1981)
7. Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. SIAM Journal on Computing, vol. 5(4) (1976)
8. Farach, M., Liberatore, V.: On local register allocation. In: 9th ACM-SIAM symposium on Discrete Algorithms, pp. 564–573. ACM Press, New York (1998)
9. Fraser, C., Hanson, D.: A Retargetable C Compiler: Design and Implementation. Addison-Wesley, Reading (1995)
10. Fu, C., Wilken, K.D.: A faster optimal register allocator. In: International Symposium on Microarchitecture, pp. 245–256. ACM Press, New York (2002)
11. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs, 2nd edn. Elsevier, Amsterdam (2004)
12. Goodwin, D.W., Wilken, K.D.: Optimal and near-optimal global register allocations using 0-1 integer programming. SPE 26(8), 929–965 (1996)
13. Hirschrott, U., Krall, A., Scholz, B.: Graph coloring vs. optimal register allocation for optimizing compilers. In: JMLC, pp. 202–213. Springer, Heidelberg (2003)
14. Kong, T., Wilken, K.D.: Precise register allocation for irregular architectures. In: International Symposium on Microarchitecture, pp. 297–307. ACM Press, New York (1998)
15. Koseki, A., Komatsu, H., Nakatani, T.: Preference-directed graph coloring. In: PLDI, pp. 297–307. ACM Press, New York (2002)
16. Nickerson, B.R.: Graph coloring register allocation for processors with multi-register operands. In: PLDI, pp. 40–52 (1990)
17. Runeson, J., Nystrom, S.-O.: Retargetable graph-coloring register allocation for irregular architectures. In: SCOPES, pp. 240–254. Springer, Heidelberg (2003)
18. Scholz, B., Eckstein, E.: Register allocation for irregular architectures. In: LCTES/SCOPES, pp. 139–148. ACM Press, New York (2002)
19. Smith, M.D., Ramsey, N., Holloway, G.: A generalized algorithm for graph-coloring register allocation. In: PLDI, pp. 277–288 (2004)

# Conservative Ambiguity Detection in Context-Free Grammars

Sylvain Schmitz

Laboratoire I3S, Université de Nice - Sophia Antipolis & CNRS, France  
schmitz@i3s.unice.fr

**Abstract.** The ability to detect ambiguities in context-free grammars is vital for their use in several fields, but the problem is undecidable in the general case. We present a safe, conservative approach, where the approximations cannot result in overlooked ambiguous cases. We analyze the complexity of the verification, and provide formal comparisons with several other ambiguity detection methods.

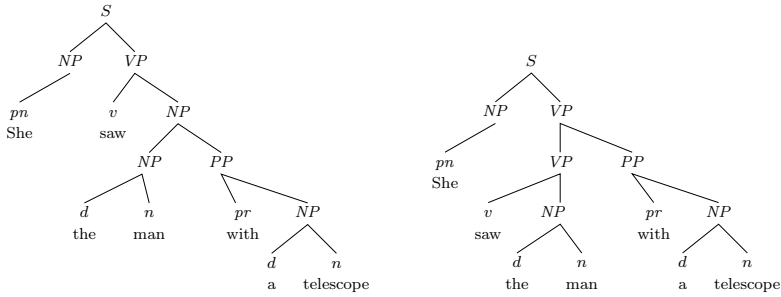
## 1 Introduction

Syntactic ambiguity allows a sentence to have more than one syntactic interpretation. A classical example is the sentence “She saw the man with a telescope.”, where the phrase “with a telescope” can be associated to “saw” or to “the man”. The presence of ambiguities in a context-free grammar (CFG) can severely hamper the reliability or the performance of the tools built from it. Sensitive fields, where CFGs are used to model the syntax, include for instance language acquisition [1], RNA analysis [2,3], controlled natural languages [4], or programming languages [5,6,7].

While proven undecidable [8,9], the problem of testing a context-free grammar for ambiguity can still be tackled approximatively. The approximations may result in two types of errors: *false negatives* if some ambiguities are left undetected, or *false positives* if some detected “ambiguities” are not actual ones.

In this paper, we present a framework for the conservative detection of ambiguities, only allowing false positives. Our general approach is that of the verification of an infinite system: we build a finite approximation of the grammar (Section 3) and check for ambiguities in this abstract structure (Section 4). The driving purpose of the paper is to establish the following theoretical results:

- An approximation model for CFGs, based on the quotienting of a graph of all the derivation trees of the grammar, which we call its *position graph*, into a nondeterministic finite automaton (NFA) (Section 3.2).
- The soundness of the verification we run on the resulting NFA. Although the ambiguity of our NFA is already a conservative test for ambiguities in the original grammar (Section 4.1), our verification improves on this immediate approach by ignoring some spurious paths (Section 4.2). The complexity of



**Fig. 1.** Two trees yielding the sentence “She saw the man with a telescope.” with  $\mathcal{G}_{\square}$

the algorithm is bounded by a quadratic function of the size of our NFA (Section 4.4).

- Formal comparisons with several ambiguity checking methods: the bounded-length detection schemes [10, 16, 11] (which are not conservative tests), the LR-Regular condition [12], and the horizontal and vertical ambiguity condition [3] (Section 5); these comparisons rely on the generality of our approximation model.

The proofs of our results can be found in a companion research report [13], and we report on the experimental results of a prototype implementation of our algorithm in a different publication [7]. Let us proceed with an overview of our approach to ambiguity detection in the coming section.

## 2 Outline

Ambiguity in a CFG is characterized as a property of its derivation trees: if two different derivation trees yield the same sentence, then we are facing an ambiguity. Considering again the classical ambiguous sentence “She saw the man with a telescope.”, a simple English grammar  $\mathcal{G}_{\square} = \langle N, T, P, S \rangle$  that presents this ambiguity could have the rules in  $P$

$$S \rightarrow NP VP, NP \rightarrow d | pn | NP PP, VP \rightarrow v NP | VP PP, PP \rightarrow pr NP, \quad (\mathcal{G}_1)$$

where the nonterminals in  $N$ , namely  $S$ ,  $NP$ ,  $VP$ , and  $PP$ , stand respectively for a sentence, a noun phrase, a verb phrase, and a preposition phrase, whereas the terminals in  $T$ , namely  $d$ ,  $n$ ,  $v$ ,  $pn$ , and  $pr$ , denote determinants, nouns, verbs, pronouns, and prepositions.<sup>1</sup> The two interpretations of our sentence are mirrored in the two derivation trees of Figure 1.

<sup>1</sup> We denote in general terminals in  $T$  by  $a, b, \dots$ , terminal strings in  $T^*$  by  $u, v, \dots$ , nonterminals by  $A, B, \dots$ , symbols in  $V = T \cup N$  by  $X, Y, \dots$ , strings in  $V^*$  by  $\alpha, \beta, \dots$ , and rules in  $P$  by  $i, j$  or by indices  $1, 2, \dots$ ;  $\varepsilon$  denotes the empty string, and  $k : x$  the prefix of length  $k$  of the string  $x$ .

### 2.1 Bracketed Grammars

Tree structures are easier to handle in a flat representation, where the structural information is described by a bracketing [14]: each rule  $i = A \xrightarrow{i} \alpha$  of the grammar is surrounded by a pair of opening and closing brackets  $d_i$  and  $r_i$ .

Formally, our *bracketed grammar* of a context-free grammar  $\mathcal{G} = \langle N, T, P, S \rangle$  is the context-free grammar  $\mathcal{G}_b = \langle N, T_b, P_b, S \rangle$  where  $T_b = T \cup T_d \cup T_r$  with  $T_d = \{d_i \mid i \in P\}$  and  $T_r = \{r_i \mid i \in P\}$ , and  $P_b = \{A \xrightarrow{i} d_i \alpha r_i \mid A \xrightarrow{i} \alpha \in P\}$ . We denote derivations in  $\mathcal{G}_b$  by  $\Rightarrow_b$ . We define the homomorphism  $h$  from  $V_b^*$  to  $V^*$  by  $h(d_i) = \varepsilon$  and  $h(r_i) = \varepsilon$  for all  $i$  in  $P$ , and  $h(X) = X$  otherwise, and denote by  $\delta_b$  (resp.  $w_b$ ) a string in  $V_b^*$  (resp.  $T_b^*$ ) such that  $h(\delta_b) = \delta$  (resp.  $h(w_b) = w$ ).

Using the rule indices as subscripts for the brackets, the two trees of Figure 1 are represented by the following two sentences of the bracketed grammar for  $\mathcal{G}'_{\square}$  [2]

$$d_1 d_2 d_4 pn r_4 d_6 v d_5 d_3 d n r_3 d_8 pr d_3 d n r_3 r_8 r_5 r_6 r_2 \$ r_1 \tag{1}$$

$$d_1 d_2 d_4 pn r_4 d_7 d_6 v d_3 d n r_3 r_6 d_8 pr d_3 d n r_3 r_8 r_7 r_2 \$ r_1. \tag{2}$$

The existence of an ambiguity can be verified by checking that the image of these two different sentences by  $h$  is the same string  $pn v d n pr d n$ .

### 2.2 Super Languages

In general, an ambiguity in a grammar  $\mathcal{G}$  is thus the existence of two different sentences  $w_b$  and  $w'_b$  of  $\mathcal{G}_b$  such that  $w = w'$ . Therefore, we can design a conservative ambiguity verification if we approximate the language  $\mathcal{L}(\mathcal{G}_b)$  with a super language and look for such sentences in the super language.

There exist quite a few methods that return a regular superset of a context-free language [15]; we present in the next section a very general model for such approximations. We can then verify on the NFA we obtain whether the original grammar might have contained any ambiguity. In Section 4 we exhibit some shortcomings of regular approximations, and present how to compute a more accurate context-free super language instead.

## 3 Position Graphs and Their Quotients

### 3.1 Position Graph

Let us consider again the two sentences (1) and (2) and how we can read them step by step on the trees of Figure 1. This process is akin to a left to right walk in the trees, between *positions* to the immediate left or immediate right of a tree node. For instance, the dot in

$$d_1 d_2 d_4 pn r_4 d_6 v d_5 d_3 d n r_3 \bullet d_8 pr d_3 d n r_3 r_8 r_5 r_6 r_2 \$ r_1 \tag{3}$$

identifies a position between  $NP$  and  $PP$  in the middle of the left tree of Figure 1.

---

<sup>2</sup> The *extended* version  $\mathcal{G}' = \langle N', T', P', S' \rangle$  of a CFG  $\mathcal{G} = \langle N, T, P, S \rangle$  adds a new start symbol  $S'$  to  $N$ , an end of sentence symbol  $\$$  to  $T$ , and a new rule  $S' \xrightarrow{1} S\$$  to  $P$ .

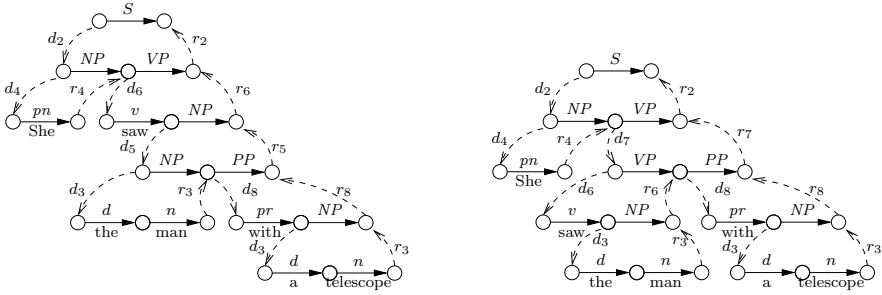


Fig. 2. Portions of the position graph of  $\mathcal{G}_{11}$  corresponding to the two trees of Figure 1

Transitions from one position to the other can then be performed upon reading the node label, upon deriving from this node, or upon returning from such a derivation. We have thus three types of transitions: symbol transitions  $\xrightarrow{X}$ , derivation transitions  $\xrightarrow{d_i}$ , and reduction transitions  $\xrightarrow{r_i}$ , where  $i$  is a rule number. The set of all these positions in all parse trees along with the transition relation is a *position graph*. Figure 2 presents two portions of the position graph for  $\mathcal{G}_{11}$ ; the position identified by the dot in (3) is now a vertex in the left graph.

Although a dotted sentence of  $\mathcal{G}_b$  like (3) suffices to identify a unique position in the derivation tree for that sentence, it is convenient to know that this position is immediately surrounded by the *NP* and *PP* symbols. We therefore denote by  $x_b d_i(\alpha \cdot \alpha'_i) r_i x'_b$  the position identified by  $x_b d_i u_b \cdot u'_i r_i x'_b$  such that the derivations

$$S' \Rightarrow_b^* x_b A x'_b \xrightarrow{i} x_b d_i \alpha \alpha'_i r_i x'_b, \quad \alpha \Rightarrow_b^* u_b \text{ and } \alpha' \Rightarrow_b^* u'_b \tag{4}$$

hold in  $\mathcal{G}'_b$ . Using this notation, the position identified by (3) is denoted by

$$d_1 d_2 d_4 pn r_4 d_6 v d_5(\overset{NP}{d_3 d n r_3} \cdot \overset{PP}{d_8 pr d_3 d n r_3 r_8}) r_5 r_6 r_2 \$ r_1. \tag{5}$$

**Definition 1.** The position graph  $\Gamma = \langle \mathcal{N}, \xrightarrow{\cdot} \rangle$  of a grammar  $\mathcal{G}$  associates the set  $\mathcal{N}$  of positions with the relation  $\xrightarrow{\cdot}$  labeled by elements of  $V_b$ , defined by

$$x_b d_i(\alpha \cdot \overset{X \alpha'}{u_b u'_b}) r_i x'_b \xrightarrow{X} x_b d_i(\overset{\alpha X}{u_b u_b} \cdot \alpha'_i) r_i x'_b \quad \text{iff } X \in V, X \Rightarrow_b^* v_b,$$

$$x_b d_i(\alpha \cdot \overset{B \alpha'}{u_b u'_b}) r_i x'_b \xrightarrow{d_j} x_b d_i u_b d_j(\cdot \overset{\beta}{u_b}) r_j u'_j r_i x'_b \quad \text{iff } B \xrightarrow{j} \beta \text{ and } \beta \Rightarrow_b^* v_b,$$

$$x_b d_i u_b d_j(\overset{\beta}{u_b} \cdot) r_j u'_j r_i x'_b \xrightarrow{r_j} x_b d_i(\overset{\alpha B}{u_b u_b} \cdot \alpha'_i) r_i x'_b \quad \text{iff } B \xrightarrow{j} \beta, \alpha \Rightarrow_b^* u_b \text{ and } \alpha' \Rightarrow_b^* u'_b.$$

We label paths in  $\Gamma$  by the sequences of labels on the individual transitions. We denote the two sets of positions at the beginning and end of the sentences by  $\mu_s = \{d_1(\cdot \overset{S \$}{u_b \$}) r_1 \mid S \Rightarrow_b^* w_b\}$  and  $\mu_f = \{d_1(\overset{S}{u_b} \cdot \$) r_1 \mid S \Rightarrow_b^* w_b\}$ . For each sentence  $w_b$  of  $\mathcal{G}_b$ , a  $\nu_s$  in  $\mu_s$  is related to a  $\nu_f$  in  $\mu_f$  by  $\nu_s \xrightarrow{S} \nu_f$ .



The parsing literature classically employs *items* to identify positions in grammars; for instance,  $[NP \xrightarrow{5} NP \cdot PP]$  is the LR(0) item [16] corresponding to position (5). There is a direct connection between these two notions: items can be viewed as equivalence classes of positions—a view somewhat reminiscent of the tree congruences of Sikkel [17].

### 3.2 Position Equivalences

In order to look for ambiguities in our grammar, we need a finite structure instead of our infinite position graph. This is provided by an equivalence relation between the positions of the graph, such that the equivalence classes become the states of a nondeterministic automaton.

**Definition 2.** *The nondeterministic position automaton  $\Gamma/\equiv$  of a context-free grammar  $\mathcal{G}$  using the equivalence relation  $\equiv$  is a tuple  $\langle Q, V'_b, R, q_s, \{q_f\} \rangle$  where*

- $Q = [\mathcal{N}]_{\equiv} \cup \{q_s, q_f\}$  is the state alphabet, where  $[\mathcal{N}]_{\equiv}$  is the set of equivalence classes  $[\nu]_{\equiv}$  over  $\mathcal{N}$  modulo the equivalence relation  $\equiv$ ,
- $V'_b$  is the input alphabet,
- $R$  in  $Q (V'_b \cup \{\varepsilon\}) \times Q$  is the set of rules  $\{q\chi \vdash q' \mid \exists \nu \in q \text{ and } \nu' \in q', \nu \xrightarrow{\chi} \nu'\} \cup \{q_s\varepsilon \vdash [\nu_s]_{\equiv} \mid \nu_s \in \mu_s\} \cup \{[\nu_f]_{\equiv} \varepsilon \vdash q_f \mid \nu_f \in \mu_f\} \cup \{q_f\$ \vdash q_f\}$ , and
- $q_s$  and  $q_f$  are respectively the initial and the final state.

If the chosen equivalence relation is of finite index, then the nondeterministic position automaton is finite. For instance, an equivalence relation that results in a NFA similar to a nondeterministic LR(0) automaton [18,19]—the main difference being the presence of the  $r_i$  transitions—is  $\text{item}_0$  defined by

$$x_b d_i \binom{\alpha}{u_b} \cdot \binom{\alpha'}{u'_b} r_i x'_b \quad \text{item}_0 \quad y_b d_j \binom{\beta}{v_b} \cdot \binom{\beta'}{v'_b} r_j y'_b \quad \text{iff } i = j \text{ and } \alpha' = \beta'. \tag{6}$$

The equivalence classes in  $[\mathcal{N}]_{\text{item}_0}$  are the LR(0) items. Figure 3 presents the nondeterministic automaton for  $\mathcal{G}_{\text{in}}$  resulting from the use of  $\text{item}_0$  as equivalence relation. Some plain  $\varepsilon$ -transitions and states of form  $\cdot A$  and  $A \cdot$  were added in order to reduce clutter in the figure. The addition of these states and transitions results in a  $\mathcal{O}(|\mathcal{G}|)$  bound on the size of  $\Gamma/\text{item}_0$  [18]. Our position (5) is now in the equivalence class represented by the state labeled by  $NP \rightarrow NP \cdot PP$ .

Let us denote by  $\models$  the relation between configurations of a NFA  $\mathcal{A} = \langle Q, \Sigma, R, q_s, F \rangle$ , such that  $qaw \models q'w$  if and only if there exists a rule  $qa \vdash q'$  in  $R$ . The language recognized by  $\mathcal{A}$  is then  $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \exists q_f \in F, q_s w \models^* q_f\}$ .

**Theorem 1.** *Let  $\mathcal{G}$  be a context-free grammar and  $\equiv$  an equivalence relation on  $\mathcal{N}$ . The language generated by  $\mathcal{G}_b$  is included in the terminal language recognized by  $\Gamma/\equiv$ , i.e.  $\mathcal{L}(\mathcal{G}_b) \subseteq \mathcal{L}(\Gamma/\equiv) \cap T_b^*$ .*

## 4 Ambiguity Detection

We are now in position to detect ambiguities on a finite, regular structure that approximates our initial grammar.

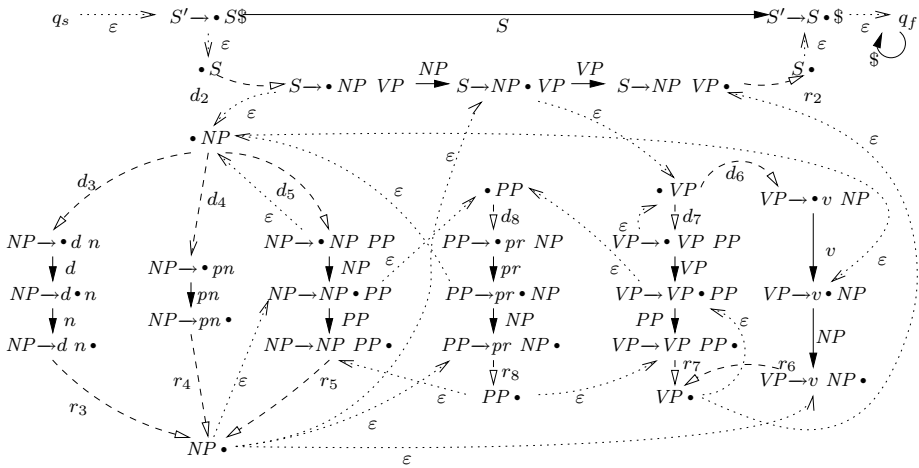


Fig. 3. The nondeterministic position automaton for  $\mathcal{G}_{\square}$  using  $\text{item}_0$

### 4.1 Regular Ambiguity Detection

Our first conservative ambiguity checking procedure relies on Theorem  $\square$ . Following the arguments developed in Section  $\square$ , an ambiguity in  $\mathcal{G}$  implies the existence of two sentences  $w_b$  and  $w'_b$  in the regular super language  $\mathcal{L}(\Gamma/\equiv) \cap T_b^*$  such that  $w = w'$ . We call a CFG with no such pair of sentences *regular*  $\equiv$  *unambiguous*, or RU( $\equiv$ ) for short.

The existence of such a pair of sentences can be tested in  $\mathcal{O}(|\Gamma/\equiv|^2)$  using accessibility relations like the ones developed in Section  $\square$ . How good is this algorithm? Being conservative is not enough for practical uses; after all, a program that always answers that the tested grammar is ambiguous is a conservative test. The regular ambiguity test sketched above performs unsatisfactorily: when using the  $\text{item}_0$  equivalence, it finds some LR(0) grammars ambiguous, like for instance  $\mathcal{G}_{\square}$  with rules

$$S \rightarrow aAa \mid bAa, A \rightarrow c. \tag{G_2}$$

The sentences  $d_2ad_4cr_4ar_2$  and  $d_2ad_4cr_4ar_3$  are both in  $\mathcal{L}(\Gamma_{\square}/\text{item}_0) \cap T_b^*$ .

The LR algorithm  $\square$  hints at a solution: we could consider nonterminal symbols in our verification and thus avoid spurious paths in the NFA. A single direct step using a nonterminal symbol represents *exactly* the context-free language derived from it, much more accurately than any regular approximation we could make for this language.

### 4.2 Common Prefixes with Conflicts

Let us consider again the two sentences  $\square$  and  $\square$ , but let us dismiss all the  $d_i$  symbols; the two sentences  $\square$  and  $\square$  we obtain are still different:

$$pn r_4 v d n r_3 pr d n r_3 r_8 r_5 r_6 r_2 \$ r_1 \tag{7}$$

$$pn r_4 v d n r_3 r_6 pr d n r_3 r_8 r_7 r_2 \$ r_1. \tag{8}$$

They share a longest common prefix  $pn r_4 v d n r_3$  before a *conflict*<sup>3</sup> between  $pr$  and  $r_6$ .

Observe that the two positions in conflict could be reached more directly in a NFA by reading the prefix  $NP v NP$ . We obtain the two sentential forms

$$NP v NP pr d n r_3 r_8 r_5 r_6 r_2 \$ r_1 \tag{9}$$

$$NP v NP r_6 pr d n r_3 r_8 r_7 r_2 \$ r_1. \tag{10}$$

We cannot however reduce our two sentences to two identical sentential forms: our common prefix with one conflict  $pn r_4 v d n r_3 r_6$  would reduce to a different prefix  $NP VP$ , and thus we do not reduce the conflicting reduction symbol  $r_6$ .

The remaining suffixes  $pr d n r_3 r_8 r_5 r_6 r_2 \$ r_1$  and  $pr d n r_3 r_8 r_7 r_2 \$ r_1$  share again a longest common prefix  $pr d n r_3 r_8$  before a conflict between  $r_5$  and  $r_7$ ; the common prefix reduces to  $PP$ , and we have the sentential forms

$$NP v NP PP r_5 r_6 r_2 \$ r_1 \tag{11}$$

$$NP v NP r_6 PP r_7 r_2 \$ r_1. \tag{12}$$

Keeping the successive conflicting reduction symbols  $r_5$ ,  $r_6$  and  $r_7$ , we finally reach a common suffix  $r_2 \$ r_1$  that cannot be reduced any further, since we need to keep our conflicting reductions. The image of our two different reduced sentential forms (11) and (12) by  $h$  is a common sentential form  $NP v NP PP \$$ , which shows the existence of an ambiguity in our grammar.

We conclude from our small example that, in order to give preference to the more accurate direct path over its terminal counterpart, we should only follow the  $r_i$  transitions in case of conflicts or in case of a common factor that cannot be reduced due to the earlier conflicts. This general behavior is also the one displayed by noncanonical parsers [20].

### 4.3 Accessibility Relations

We implement the idea of common prefixes with conflicts in the mutual accessibility relations classically used to find common prefixes [21, Chapter 10]. Mutual accessibility relations are used to identify couples of states accessible upon reading the same language from the starting couple  $(q_s, q_s)$ , which brings the complexity of the test down to a quadratic function in the number of transitions, and avoids the potential exponential blowup of a NFA determinization.

The case where reduction transitions should be followed after a conflict is handled by considering pairs over  $\mathbb{B} \times Q$  instead of  $Q$ : the boolean tells whether we followed a  $d_i$  transition since the last conflict. In order to improve readability, we write  $q\chi \vdash q'$  for  $q$  and  $q'$  in  $\mathbb{B} \times Q$  if their states allow this transition to occur. The predicate  $\downarrow q$  in  $\mathbb{B}$  denotes that we are allowed to ignore a reduction transition. Our starting couple  $(q_s, q_s)$  has its boolean values initially set to true.

---

<sup>3</sup> Our notion of conflict coincides with that of LR(0) conflicts when one employs `item0`.

**Definition 3.** *The primitive mutual accessibility relations over  $(\mathbb{B} \times Q)^2$  are*

**shift.** *mas defined by  $(q_1, q_2)$  mas  $(q_3, q_4)$  if and only if there exists  $X$  in  $V$  such that  $q_1X \vdash q_3$  and  $q_2X \vdash q_4$*

**epsilon.** *mae=mael  $\cup$  maer where  $(q_1, q_2)$  mael  $(q_3, q_2)$  if and only if  $q_1d_i \vdash q_3$  or  $q_1\varepsilon \vdash q_3$  and  $\setminus q_3$  and symmetrically for maer,  $(q_1, q_2)$  maer  $(q_1, q_4)$  if and only if  $q_2d_i \vdash q_4$  or  $q_2\varepsilon \vdash q_4$ , and  $\setminus q_4$ ,*

**reduction.** *mar defined by  $(q_1, q_2)$  mar  $(q_3, q_4)$  if and only if there exists  $i$  in  $P$  such that  $q_1r_i \vdash q_3$  and  $q_2r_i \vdash q_4$ , and furthermore  $\neg \setminus q_1$  or  $\neg \setminus q_2$ , and then  $\neg \setminus q_3$  and  $\neg \setminus q_4$ ,*

**conflict.** *mac=macl  $\cup$  macr with  $(q_1, q_2)$  macl  $(q_3, q_2)$  if and only if there exist  $i$  in  $P$ ,  $q_4$  in  $Q$  and  $z$  in  $T_d^* \cdot T'$  such that  $q_1r_i \vdash q_3$ ,  $q_2z \models^+ q_4$  and  $\neg \setminus q_3$ , and symmetrically for macr,  $(q_1, q_2)$  macr  $(q_1, q_4)$  if and only if there exist  $i$  in  $P$ ,  $q_3$  in  $Q$  and  $z$  in  $T_d^* \cdot T'$  such that  $q_2r_i \vdash q_4$ ,  $q_1z \models^+ q_3$ , and  $\neg \setminus q_4$ .*

*The global mutual accessibility relation ma is defined as mas  $\cup$  mae  $\cup$  mar  $\cup$  mac.*

These relations are akin to the item construction of a LR parser: the relation mas corresponds to a shift, the relation mae to an item closure, the relation mar to a goto, and the relation mac to a LR conflict.

Let us call a grammar  $\mathcal{G}$  such that  $(q_s, q_s)$  (mae  $\cup$  mas) $^* \circ$  mac  $\circ$  ma $^*$   $(q_f, q_f)$  does not hold in  $\Gamma/\equiv$  noncanonically  $\equiv$ -unambiguous, or NU( $\equiv$ ) for short.

**Theorem 2.** *Let  $\mathcal{G}$  be a context-free grammar and  $\equiv$  a position equivalence relation. If  $\mathcal{G}$  is ambiguous, then  $\mathcal{G}$  is not NU( $\equiv$ ).*

### 4.4 Complexity

The complexity of our algorithm depends mostly on the equivalence relation we choose to quotient the position graph. Supposing that we choose an equivalence relation  $\equiv$  of finite index and of decidable computation of complexity  $\mathcal{C}(\Gamma/\equiv)$ , then we need to build the image ma $^*$   $(\{(q_s, q_s)\})$ . This step and the search for a conflict in this image can both be performed in time  $\mathcal{O}(|\Gamma/\equiv|^2)$ . The overall complexity of our algorithm is thus  $\mathcal{O}(\mathcal{C}(\Gamma/\equiv) + |\Gamma/\equiv|^2)$ .

The complexity  $\mathcal{C}(\Gamma/\text{item}_0)$  of the construction of the collapsed position graph  $\Gamma/\text{item}_0$  is linear with the size of the resulting nondeterministic position automaton. The overall complexity of our ambiguity detection algorithm when one uses item $_0$  is therefore  $\mathcal{O}(|\mathcal{G}|^2)$ .

## 5 Formal Comparisons

We compare here our ambiguity detection algorithm with some of the other means to test a context-free grammar for ambiguity we are aware of. We first establish the edge of our algorithm over the regular ambiguity test of Section 4.1. The comparison with LR-Regular testing requires the full power of our method, and at last, the horizontal and vertical ambiguity detection technique is shown to be incomparable with our own.

### 5.1 Regular Ambiguity

Theorem 3, along with the example of  $\mathcal{G}_2$ , shows a strict improvement of our method over the simple algorithm discussed in Section 4.1.

**Theorem 3.** *If  $\mathcal{G}$  is  $RU(\equiv)$ , then it is also  $NU(\equiv)$ .*

### 5.2 Bounded Length Detection Schemes

Many algorithms specifically designed for ambiguity detection look for ambiguities in all sentences up to some length [10,16,11]. As such, they fail to detect ambiguities beyond that length: they allow false negatives. Nonetheless, these detection schemes can vouch for the ambiguity of any string shorter than the given length; this is valuable in applications where, in practice, the sentences are of a small bounded length. The same guarantee is offered by the equivalence relation  $\text{prefix}_m$  defined for any fixed length  $m$  by<sup>4</sup>

$$x_b d_i \left( \begin{smallmatrix} \alpha \\ u_b \end{smallmatrix} \cdot \begin{smallmatrix} \alpha' \\ u'_b \end{smallmatrix} \right) r_i x'_b \text{ prefix}_m y_b d_j \left( \begin{smallmatrix} \beta \\ v_b \end{smallmatrix} \cdot \begin{smallmatrix} \beta' \\ v'_b \end{smallmatrix} \right) r_j y'_b \text{ iff } m :_b x_b u_b = m :_b y_b v_b. \quad (13)$$

Provided that  $\mathcal{G}$  is acyclic,  $\Gamma/\text{prefix}_m$  is finite.

**Theorem 4.** *Let  $w_b$  and  $w'_b$  be two bracketed sentences in  $\mathcal{L}(\Gamma/\text{prefix}_m) \cap T_b^*$  with  $w = w'$  and  $|w| \leq m$ . Then  $w_b$  and  $w'_b$  are in  $\mathcal{L}(\mathcal{G}_b)$ .*

Outside of the specific situation of languages that are finite in practice, bounded length detection schemes can be quite costly to use. The performance issue can be witnessed with the two families of grammars  $\mathcal{G}_3^n$  and  $\mathcal{G}_4^n$  with rules

$$S \rightarrow A | B_n, A \rightarrow Aaa | a, B_1 \rightarrow aa, B_2 \rightarrow B_1 B_1, \dots, B_n \rightarrow B_{n-1} B_{n-1} \quad (\mathcal{G}_3^n)$$

$$S \rightarrow A | B_n a, A \rightarrow Aaa | a, B_1 \rightarrow aa, B_2 \rightarrow B_1 B_1, \dots, B_n \rightarrow B_{n-1} B_{n-1}, \quad (\mathcal{G}_4^n)$$

where  $n \geq 1$ . In order to detect the ambiguity of  $\mathcal{G}_4^n$ , a bounded length algorithm would have to explore all strings in  $\{a\}^*$  up to length  $2^n + 1$ . Our algorithm correctly finds  $\mathcal{G}_3^n$  unambiguous and  $\mathcal{G}_4^n$  ambiguous in time  $\mathcal{O}(n^2)$  using  $\text{item}_0$ .

### 5.3 LR(k) and LR-Regular Testing

Conservative algorithms do exist in the programming language parsing community, though they are not primarily meant as ambiguity tests. Nonetheless, a full LALR or LR construction is often used as a practical test for non ambiguity [2]. The LR(k) testing algorithms [16,18,19] are much more efficient in the worst case and provided our initial inspiration. Our position automaton is a generalization of the item grammar or nondeterministic automaton of these works, and our test looks for ambiguities instead of LR conflicts. Let us consider again  $\mathcal{G}_3$ : it requires a LR( $2^n$ ) test for proving its unambiguity, but it is simply  $NU(\text{item}_0)$ .

---

<sup>4</sup> The *bracketed prefix*  $m :_b x_b$  of a bracketed string  $x_b$  is defined as the longest string in  $\{y_b \mid x_b = y_b z_b \text{ and } |y| = m\}$  if  $|x| > m$  or simply  $x_b$  if  $|x| \leq m$ .

One of the strongest ambiguity tests available is the LR-Regular condition [12,22]: instead of merely checking the  $k$  next symbols of lookahead, a LRR parser considers regular equivalence classes on the entire remaining input to infer its decisions. Given  $\Pi$  a finite regular partition of  $T^*$  that defines a left congruence  $\cong$ , a grammar  $\mathcal{G}$  is LR( $\Pi$ ) if and only if  $S \xrightarrow{\text{rm}}^* \delta Ax \xrightarrow{\text{rm}} \delta \alpha x$ ,  $S \xrightarrow{\text{rm}}^* \gamma By \xrightarrow{\text{rm}} \gamma \beta y = \delta \alpha z$  and  $x \cong z \pmod{\Pi}$  imply  $A \rightarrow \alpha = B \rightarrow \beta$ ,  $\delta = \gamma$  and  $y = z$ .

Our test for ambiguity is strictly stronger than the LR( $\Pi$ ) condition with the equivalence relation  $\text{item}_{\Pi} = \text{item}_0 \cap \text{look}_{\Pi}$ , where  $\text{look}_{\Pi}$  is defined by

$$x_b d_i(\overset{\alpha}{u_b} \cdot \overset{\alpha'}{u'_b}) r_i x'_b \text{ look}_{\Pi} y_b d_j(\overset{\beta}{v_b} \cdot \overset{\beta'}{v'_b}) r_j y'_b \text{ iff } u'x' \cong v'y' \pmod{\Pi}. \tag{14}$$

**Theorem 5.** *If  $\mathcal{G}$  is LR( $\Pi$ ), then it is also NU( $\text{item}_{\Pi}$ ).*

Let us consider now the grammar with rules

$$S \rightarrow AC \mid BCb, \quad A \rightarrow a, \quad B \rightarrow a, \quad C \rightarrow cCb \mid cb. \tag{G_5}$$

Grammar  $\mathcal{G}_5$  is not LRR: the right contexts  $c^n b^n \$$  and  $c^n b^{n+1} \$$  of the reductions using  $A \rightarrow a$  and  $B \rightarrow a$  cannot be distinguished by regular covering sets. Nevertheless, our test on  $\Gamma_5 / \text{item}_0$  shows that  $\mathcal{G}_5$  is not ambiguous

### 5.4 Horizontal and Vertical Ambiguity

Brabrand *et al.* [3] recently proposed an ambiguity detection scheme also based on regular approximations of the grammar language. Its originality lies in the decomposition of the ambiguity problem into two (also undecidable) problems, namely the horizontal and vertical ambiguity problems. The detection method then relies on the fact that a context-free grammar is unambiguous if and only if it is horizontal and vertical unambiguous. The latter tests are performed on a regular approximation of the grammar [23].

**Definition 4.** *The automaton  $\Gamma / \equiv$  is vertically ambiguous if and only if there exist an  $A$  in  $N$  with two different productions  $A \xrightarrow{i} \alpha_1$  and  $A \xrightarrow{j} \alpha_2$ , and the bracketed strings  $x_b, x'_b, u_b, u'_b, w_b$ , and  $w'_b$  in  $T_b^*$  with  $w = w'$  such that*

$$[x_b d_i(\overset{\alpha_1}{u_b} \cdot \overset{\alpha_1'}{u'_b}) r_i x'_b]_{\equiv} w_b \vDash^* [x_b d_i(\overset{\alpha_1}{u_b} \cdot \overset{\alpha_1'}{u'_b}) r_i x'_b]_{\equiv} \text{ and}$$

$$[x_b d_j(\overset{\alpha_2}{u'_b} \cdot \overset{\alpha_2'}{u'_b}) r_j x'_b]_{\equiv} w'_b \vDash^* [x_b d_j(\overset{\alpha_2}{u'_b} \cdot \overset{\alpha_2'}{u'_b}) r_j x'_b]_{\equiv}.$$

*The automaton  $\Gamma / \equiv$  is horizontally ambiguous if and only if there is a production  $A \xrightarrow{i} \alpha$  in  $P$ , a decomposition  $\alpha = \alpha_1 \alpha_2$ , and the bracketed strings  $x_b, x'_b, u_b, u'_b, v_b, v'_b, w_b, w'_b$ , and  $y_b$  in  $T_b^*$  with  $v = v'$  and  $w = w'$  such that*

$$[x_b d_i(\overset{\alpha_1 \alpha_2}{u_b u'_b} \cdot \overset{\alpha_1 \alpha_2'}{u'_b u'_b}) r_i x'_b]_{\equiv} v_b y_b w_b \vDash^* [x_b d_i(\overset{\alpha_1}{u_b} \cdot \overset{\alpha_2}{u'_b}) r_i x'_b]_{\equiv} y_b w_b \vDash^* [x_b d_i(\overset{\alpha_1 \alpha_2}{u_b u'_b} \cdot \overset{\alpha_1 \alpha_2'}{u'_b u'_b}) r_i x'_b]_{\equiv}$$

$$[x_b d_i(\overset{\alpha_1 \alpha_2}{u_b u'_b} \cdot \overset{\alpha_1 \alpha_2'}{u'_b u'_b}) r_i x'_b]_{\equiv} v'_b y_b w'_b \vDash^* [x_b d_i(\overset{\alpha_1}{u_b} \cdot \overset{\alpha_2}{u'_b}) r_i x'_b]_{\equiv} w'_b \vDash^* [x_b d_i(\overset{\alpha_1 \alpha_2}{u_b u'_b} \cdot \overset{\alpha_1 \alpha_2'}{u'_b u'_b}) r_i x'_b]_{\equiv}.$$

**Theorem 6.** *Let  $\mathcal{G}$  be a context-free grammar and  $\Gamma/\equiv$  its position automaton. If  $\mathcal{G}$  is  $RU(\equiv)$ , then  $\Gamma/\equiv$  is horizontally and vertically unambiguous.*

Theorem 6 shows that the horizontal and vertical ambiguity criteria result in a better conservative ambiguity test than regular  $\equiv$ -ambiguity, although at a higher price:  $\mathcal{O}(|\mathcal{G}|^5)$  in the worst case. Owing to these criteria, the technique of Brabrand *et al.* accomplishes to show that the palindrome grammar with rules

$$S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon \quad (\mathcal{G}_6)$$

is unambiguous, which seems impossible with our scheme. On the other hand, even when they employ *unfolding* techniques, they are always limited to regular approximations, and fail to see that the LR(0) grammar with rules

$$S \rightarrow AA, A \rightarrow aAa \mid b \quad (\mathcal{G}_7)$$

is unambiguous. The two techniques are thus incomparable, and could benefit from each other.

## 6 Conclusion

As a classical undecidable problem in formal languages, ambiguity detection in context-free grammars did not receive much practical attention. Nonetheless, the ability to provide a conservative test could be applied in many fields where context-free grammars are used. This paper presents one of the few conservative tests explicitly aimed towards ambiguity checking, along with the recent work of Brabrand *et al.* [3].

The ambiguity detection scheme we presented here provides some insights on how to tackle undecidable problems on approximations of context-free languages. The general method can be applied to different decision problems, and indeed has also been put to work in the construction of an original parsing method [24] where the amount of lookahead needed is not preset but computed for each parsing decision. We hope to see more applications of this model in the future.

*Acknowledgements.* The author is highly grateful to Jacques Farré for his invaluable help at all stages of the preparation of this work. The author also thanks the anonymous referees for their numerous helpful remarks.

## References

1. Cheung, B.S.N., Uzgalis, R.C.: Ambiguity in context-free grammars. In: SAC'95, pp. 272–276. ACM Press, New York (1995), doi:10.1145/315891.315991
2. Reeder, J., Steffen, P., Giegerich, R.: Effective ambiguity checking in biosequence analysis. BMC Bioinformatics 6, 153 (2005)
3. Brabrand, C., Giegerich, R., Møller, A.: Analyzing ambiguity of context-free grammars. Technical Report RS-06-09, BRICS, University of Aarhus (May 2006)
4. AeroSpace and Defence Industries Association of Europe: ASD Simplified Technical English, Specification ASD-STE100 (2005)

5. Kuich, W.: Systems of pushdown acceptors and context-free grammars. *Elektronische Informationsverarbeitung und Kybernetik* 6(2), 95–114 (1970)
6. Schröder, F.W.: AMBER, an ambiguity checker for context-free grammars. Technical report, [compilertools.net](http://compilertools.net) (2001)
7. Schmitz, S.: An experimental ambiguity detection tool. In: Sloane, A., Johnstone, A., eds.: *LDTA'07*, To appear in *Electronic Notes in Theoretical Computer Science* (2007)
8. Cantor, D.G.: On the ambiguity problem of Backus systems. *Journal of the ACM* 9(4), 477–479 (1962)
9. Chomsky, N., Schützenberger, M.P.: The algebraic theory of context-free languages. In: Braffort, P., Hirshberg, D. (eds.) *Computer Programming and Formal Systems. Studies in Logic*, pp. 118–161. North-Holland Publishing, Amsterdam (1963)
10. Gorn, S.: Detection of generative ambiguities in context-free mechanical languages. *Journal of the ACM* 10(2), 196–208 (1963)
11. Jampana, S.: Exploring the problem of ambiguity in context-free grammars. Master's thesis, Oklahoma State University (July 2005)
12. Culik, K., Cohen, R.: LR-Regular grammars—an extension of LR( $k$ ) grammars. *Journal of Computer and System Sciences* 7, 66–96 (1973)
13. Schmitz, S.: Conservative ambiguity detection in context-free grammars. Technical Report I3S/RR-2006-30-FR, Laboratoire I3S, Université de Nice - Sophia Antipolis & CNRS (September 2006)
14. Ginsburg, S., Harrison, M.A.: Bracketed context-free languages. *Journal of Computer and System Sciences* 1, 1–23 (1967)
15. Nederhof, M.J.: Regular approximation of CFLs: a grammatical view. In: Bunt, H., Nijholt, A. (eds.) *Advances in Probabilistic and other Parsing Technologies*, pp. 221–241. Kluwer Academic Publishers, Boston, MA (2000)
16. Knuth, D.E.: On the translation of languages from left to right. *Information and Control* 8(6), 607–639 (1965)
17. Sikkel, K. (ed.): *Parsing Schemata - a framework for specification and analysis of parsing algorithms*. Texts in Theoretical Computer Science - An EATCS Series. Springer, Heidelberg (1997)
18. Hunt III, H.B., Szymanski, T.G., Ullman, J.D.: Operations on sparse relations and efficient algorithms for grammar problems. In: *15th Annual Symposium on Switching and Automata Theory*, pp. 127–132. IEEE Computer Society, Los Alamitos (1974)
19. Hunt III, H.B., Szymanski, T.G., Ullman, J.D.: On the complexity of LR( $k$ ) testing. *Communications of the ACM* 18(12), 707–716 (1975), doi:10.1145/361227.361232
20. Szymanski, T.G., Williams, J.H.: Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing* 5(2), 231–250 (1976), doi:10.1137/0205019
21. Sippu, S., Soisalon-Soininen, E.: *Parsing Theory, Vol. II: LR( $k$ ) and LL( $k$ ) Parsing*. In: *Simple Program Schemes and Formal Languages. LNCS, vol. 20*, Springer, Heidelberg (1990)
22. Heilbrunner, S.: Tests for the LR-, LL-, and LC-Regular conditions. *Journal of Computer and System Sciences* 27(1), 1–13 (1983)
23. Mohri, M., Nederhof, M.J.: Regular approximations of context-free grammars through transformation. In: Junqua, J.C., van Noord, G. (eds.) *Robustness in Language and Speech Technology*, pp. 153–163. Kluwer Academic Publishers, Dordrecht (2001)
24. Gálvez, J.F., Schmitz, S., Farré, J.: Shift-resolve parsing: Simple, linear time, unbounded lookahead. In: Ibarra, O.H., Yen, H.-C. (eds.) *CIAA 2006. LNCS, vol. 4094*, pp. 253–264. Springer, Heidelberg (2006)



# Lower Bounds for Quantile Estimation in Random-Order and Multi-pass Streaming

Sudipto Guha<sup>1,\*</sup> and Andrew McGregor<sup>2</sup>

<sup>1</sup> University of Pennsylvania  
sudipto@cis.upenn.edu

<sup>2</sup> University of California, San Diego  
andrewm@ucsd.edu

**Abstract.** We present lower bounds on the space required to estimate the quantiles of a stream of numerical values. Quantile estimation is perhaps the most studied problem in the data stream model and it is relatively well understood in the basic single-pass data stream model in which the values are ordered adversarially. Natural extensions of this basic model include the *random-order model* in which the values are ordered randomly (e.g. [21,5,13,11,12]) and the *multi-pass model* in which an algorithm is permitted a limited number of passes over the stream (e.g. [6,7,11,9,2,6,7,11,9,2]). We present lower bounds that complement existing upper bounds [21,11] in both models. One consequence is an exponential separation between the random-order and adversarial-order models: using  $\Omega(\text{polylog } n)$  space, exact selection requires  $\Omega(\log n)$  passes in the adversarial-order model while  $O(\log \log n)$  passes are sufficient in the random-order model.

## 1 Introduction

One of the principal theoretical motivations for studying the data stream model is to understand the impact of the order of the input on computation. While an algorithm in the RAM model can process the input data in an arbitrary order, a key constraint of the data stream model is that any algorithm must process (in small space) the input data in the order in which it arrives. Parameterizing the number of passes that an algorithm may have over the data establishes a spectrum between the RAM model and the one-pass data stream model. How does the computational power of the model change along this spectrum? Furthermore, what role is played by the ordering of the stream?

These issues date back to one of the earliest papers on the data stream model in which Munro and Paterson considered the problems of sorting and selection in limited space [21]. They showed that  $\tilde{O}(n^{1/p})$  space was sufficient to find the exact median of a sequence of  $n$  numbers given  $p$  passes over the data. However, if the data was randomly ordered,  $\tilde{O}(n^{1/(2p)})$  space sufficed. They

---

\* This research was supported by in part by an Alfred P. Sloan Research Fellowship and by NSF Awards CCF-0430376, and CCF-0644119.

also showed lower bounds for deterministic algorithms that stored the stream values as indivisible objects and uses a comparison based model. Specifically, they showed that all such algorithms required  $\Omega(n^{1/p})$  space in the adversarial-order model and that single-pass algorithms that maintain a set of “elements whose ranks among those read thus far are consecutive and as close to the current median as possible” require  $\Omega(\sqrt{n})$  space in the random-order model. They also conjectured the existence of an algorithm in the random-order model that used  $O(\log \log n)$  passes and  $O(\text{polylog } n)$  space to compute the median exactly. Median finding or quantile estimation has since become one of the most extensively studied problems in the data stream model [17,18,10,9,14,4,23,3]. However, it was only recently shown that there does indeed exist an algorithm which uses  $O(\log \log n)$  passes and  $O(\text{polylog } n)$  space in the random-order model [11]. This result was based on a single-pass algorithm in the random-order model that, with high probability, returned an element of rank  $n/2 \pm O(n^{1/2+\epsilon})$  and used  $\text{poly}(\epsilon^{-1}, \log n)$  space. In contrast, any algorithm in the adversarial-order model requires  $\Omega(n^{1-\delta})$  space to find an element of rank  $n/2 \pm n^\delta$ . These two facts together showed that the random-order model is strictly more powerful than the adversarial-order model.

Based on the algorithms of Munro and Paterson, it seemed plausible that any  $p$  pass algorithm in the random order stream model can be simulated by a  $2p$  pass algorithm in the adversarial streaming model. This was conjectured by Kannan [15]. Further support for this conjecture came via work initiated by Feigenbaum et al. [8] that considered the relationship between various property testing models and the data-stream model. It was shown in Guha et al. [13] that several models of property testing can be simulated in the single-pass random-order stream model while it appeared that a similar simulation in the adversarial-model required two passes. While this appeared to support the conjecture, the conjecture remained unresolved.

In this paper we show that the conjecture is false. In fact, the separation between the random-order model and the adversarial-order model can be exponential. We show that using  $p$  passes,  $\Omega(n^{1/p} p^{\Theta(1)})$ -space is required to compute the median exactly. This is a fully general lower bound as opposed to the lower bound for a restricted class of algorithms presented in [21]. Our proof is information-theoretic and uses a reduction from the communication complexity of a generalized form of pointer-chasing for which we prove the first lower-bound. It is also possible to establish a weaker lower bound using our reduction combined with the round-elimination lemma of Miltersen et al. [20] or the standard form of pointer-chasing considered by Nisan and Widgerson [22] as opposed our new lower bound for generalized pointer-chasing. We omit the details but stress that our communication complexity result for generalized pointer chasing is necessary to prove the stronger bound. Furthermore, we believe that this result may be useful for obtaining improved lower-bounds for other streaming problems.

A final question is whether it is possible to significantly improve upon the algorithm presented in [11] for the random-order model. In particular, does there exist a one-pass sub-polynomial approximation in  $O(\text{polylog } n)$ -space? We show

that this is not the case and, in particular, a single-pass algorithm returning the exact median requires  $\Omega(\sqrt{n})$  space in the random-order model. This result is about fully general algorithms in contrast to the result by Munro and Paterson [21]. We note that this is the first unqualified lower bound in the random-order model. The proof uses a reduction from communication complexity but deviates significantly from the usual form of such reductions because of the novel challenges arising when proving a lower bound in the random-order model as opposed to the adversarial-model.

## 1.1 Summary of Results and Overview

Our two main results of this paper are lower-bounds for approximate median finding in the random-order stream model and the multi-pass stream models.

In Section 3, we prove that any algorithm that returns an  $n^\delta$ -approximate median of a randomly ordered stream with probability at least  $3/4$  requires  $\Omega(\sqrt{n^{1-3\delta}/\log n})$  space. This rules out sub-polynomial approximation using poly-logarithmic space.

In Section 4, we prove that any algorithm that returns an  $n^\delta$ -approximate median in  $k$  passes of an adversarially ordered stream requires  $\Omega(n^{(1-\delta)/k} k^{-6})$  space. This disproves the conjecture that stated that any problem that could be solved in  $k/2$  passes of a randomly ordered stream could be solved in at most  $k$  passes of an adversarially ordered stream [15].

We also simplify and improve the upper bound in [11] and show that there exists a single pass algorithm using  $O(1)$  words of space that, given any  $k$ , returns an element of rank  $k \pm O(k^{1/2} \log^2 k)$  if the stream is randomly ordered. This represents an improvement in terms of space use and accuracy. However, this improvement is not the focus of the paper and can be found in Appendix A.

## 2 Preliminaries

We start by clarifying the definition of an approximate quantile of a multi-set.

**Definition 1 (Rank and Approximate Selection).** *The rank of an item  $x$  in a set  $S$  is defined as,  $\text{RANK}_S(x) = |\{x' \in S | x' < x\}| + 1$ . Assuming there are no duplicate elements in  $S$ , we say  $x$  is an  $\Upsilon$ -approximate  $k$ -rank element in  $S$  if,  $\text{RANK}_S(x) = k \pm \Upsilon$ . If there are duplicate elements in  $S$  then we say  $x$  is an  $\Upsilon$ -approximate  $k$ -rank element if there exists some way of ordering identical elements such that  $x$  is an  $\Upsilon$ -approximate  $k$ -rank element.*

## 3 Random Order Lower-Bound

In this section we will prove a lower bound of the space required to  $n^\delta$ -approximate the median in a randomly ordered stream. Our lower-bound will be based on a reduction from the communication complexity of indexing [16]. However, the reduction is significantly more involved than typical reductions because different

segments of a stream can not be determined independently by different players if the stream is in random order.

Let Alice have a binary string  $\sigma$  of length  $s' = \epsilon n^{-\delta} \sqrt{n_2} / (100 \ln(2/\epsilon))$  and let Bob have an index  $r \in [s']$  where  $\epsilon$  and  $n_2$  will be specified shortly. It is known that for Bob to learn  $\sigma_r$  with probability at least  $3/4$  after a single message from Alice then the message Alice sends must be  $\Omega(s')$  bits. More precisely,

**Theorem 1.** *There exists a constant  $c^*$  such that  $R_{1/4}^1(\text{INDEX}) \geq c^* s'$ .*

The basic idea of our proof is that if there exists an algorithm  $\mathcal{A}$  that computes the median of a randomly ordered stream in a single pass then this gives rise to a 1-way communication protocol that solves INDEX. The protocol is based upon simulating  $\mathcal{A}$  on a stream of length  $n$  where Alice determines the first  $n_1 = n - c^* n^{1-\delta} / (4 \log n)$  elements and Bob determines the remaining  $n_2 = c^* n^{1-\delta} / (4 \log n)$  elements. The stream consists of the following sets of elements:

1.  $S$ : A set of  $s = n^\delta s'$  elements  $\bigcup_{j \in [n^\delta]} \{2i + \sigma_i : i \in [s']\}$ . Note that each of the  $s'$  distinct elements occurs  $n^\delta$  times. We refer to  $S$  as the “special” elements.
2.  $X$ :  $x = (n + 1)/2 - r$  copies of 0.
3.  $Y$ :  $y = (n - 1)/2 - s + r$  copies of  $2s + 2$ .

Note that any  $n^\delta$ -approximate median of  $U = S \cup X \cup Y$  is  $2r + \sigma_r$ .

The difficulty in the proof comes from the fact that the probability that  $\mathcal{A}$  finds an  $n^\delta$ -approximate median depends on the random ordering of the stream. Hence, it would seem that Alice and Bob need to ensure that the ordering of  $U$  in the stream is chosen at random. Unfortunately that is not possible without excessive communication between Alice and Bob. Instead we will show that it is possible for Alice and Bob to generate a stream in “semi-random” order according to the following notion of semi-random.

**Definition 2 ( $\epsilon$ -Generated Random Order).** *Consider a set of elements  $\{x_1, \dots, x_n\}$ . Then  $\sigma \in \text{Sym}_n$  defines a stream  $\langle x_{\sigma(1)}, \dots, x_{\sigma(n)} \rangle$  where  $\text{Sym}_n$  is the set of all permutations on  $[n]$ . We say the ordering of this stream is  $\epsilon$ -Generated Random if  $\sigma$  is chosen according to some distribution  $\nu$  such that  $\|\mu - \nu\|_1 \geq \epsilon$  where  $\mu$  is the uniform distribution over all possible orderings.*

The importance of this definition is captured in the following simple lemma.

**Lemma 1.** *Let  $\mathcal{A}$  be a randomized algorithm that estimates some property of a randomly ordered stream such that the estimate satisfies some guarantee with probability at least  $p$ . Then the estimate returned by running  $\mathcal{A}$  on a stream in  $\epsilon$ -generated random order satisfies the same guarantees with probability at least  $p - \epsilon$ .*

*Proof.* We say the  $\mathcal{A}$  succeeds if the estimate returns satisfies the required guarantees. Let  $\Pr_{\mu, \text{coin}}(\cdot)$  denote the probability of an event over the internal coin tosses of  $\mathcal{A}$  and the ordering of the stream when the stream order is chosen according to distribution  $\mu$ . Similarly define  $\Pr_{\nu, \text{coin}}(\cdot)$  where  $\nu$  is any distribution satisfying  $\|\mu - \nu\|_1 \leq \epsilon$ .

$$\Pr_{\mu, \text{coin}} (\mathcal{A} \text{ succeeds}) = \sum_{\sigma \in \text{Sym}_n} \Pr_{\mu} (\sigma) \Pr_{\text{coin}} (\mathcal{A} \text{ succeeds} | \sigma) \leq \Pr_{\nu, \text{coin}} (\mathcal{A} \text{ succeeds}) + \epsilon .$$

Consequently, if we can show that Alice and Bob can generate a stream that is in  $O(\epsilon)$ -generation random order then by appealing to Lemma 1 we can complete the proof.

Let  $A$  be a set of  $n_1$  elements in  $U$  and  $B = U \setminus A$  be a set of  $n_2$  elements.  $A$  will be chosen randomly according to one of two distributions. We consider the following families of events.

$$E_t = \{a : |A \cap X| = |A \cap Y| + t\} \text{ and } S_{s_1} = \{a : |A \cap S| = s_1\} .$$

We define two distributions  $\mu$  and  $\mu'$ . Let  $\mu$  be the distribution where  $A$  is chosen uniformly at random from all subsets of size  $n_1$  of  $U$ . Note that,

$$\begin{aligned} \Pr_{\mu} (S_{s_1}) &= \binom{n_1}{s_1} \binom{n_2}{s_2} / \binom{n}{s} \\ \Pr_{\mu} (E_t | S_{s_1}) &= \binom{n_1 - s_1}{(n_1 - s_1)/2 - t/2} \binom{n_2 - s_2}{(n_2 - s_2)/2 + t/2} / \binom{n - s}{x} \\ \Pr_{\mu} (\{a\} | E_t, S_{s_1}) &= \begin{cases} \frac{1}{|E_t \cap S_{s_1}|} & \text{if } a \in E_t \cap S_{s_1} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where  $s_1 + s_2 = s$ . Note that the above three equations fully specify  $\mu$  since

$$\Pr_{\mu} (\{a\}) = \sum_{t, s_1} \Pr_{\mu} (\{a\} | E_t, S_{s_1}) \Pr_{\mu} (E_t | S_{s_1}) \Pr_{\mu} (S_{s_1}) .$$

Let  $\mu'$  be a distribution on  $A$  where  $\Pr_{\mu'} (S_{s_1}) = \Pr_{\mu} (S_{s_1})$ ,  $\Pr_{\mu'} (\{a\} | E_t, S_{s_1}) = \Pr_{\mu} (\{a\} | E_t, S_{s_1})$  and,

$$\Pr_{\mu'} (E_t | S_{s_1}) = \binom{n_1 - s_1}{(n_1 - s_1)/2 - t/2} \binom{n_2 - s_2}{(n_2 - s_2)/2 + t/2} / \binom{n - s}{(n - s)/2}$$

where  $s_1 + s_2 = s$ . Note that  $\mu' = \mu$  if  $r = s/2$ . The crux of the proofs is that  $\mu$  and  $\mu'$  are closely related even if  $r$  is as small as 1 or as large as  $s$ .

**Lemma 2.** *If  $s_1 \leq \frac{\epsilon \sqrt{n_2}}{100 \ln(2/\epsilon)}$  and  $t < t^*$  where  $t^* = \sqrt{2n_2 \ln(2/\epsilon)} + s$  then,*

$$\frac{1}{1 + \epsilon} \leq \frac{\Pr_{\mu} (E_t | S_{s_1})}{\Pr_{\mu'} (E_t | S_{s_1})} \leq 1 + \epsilon .$$

We omit the proof of this lemma and subsequent lemmas whose proofs, while detailed, do not require any non-standard ideas. Next, we ascertain that it is sufficient to consider only values of  $t < t^*$ .

**Lemma 3.**  *$E^* := \bigcup_{|t| < t^*} E_t$  is a high probability event under  $\mu'$  and  $\mu$ , i.e.,  $\min(\Pr_{\mu} (E^*), \Pr_{\mu'} (E^*)) \geq 1 - \epsilon$ .*

Let  $S^{**}$  be the event that the number of distinct special items in the suffix of the stream is at most  $s^{**} := c^*s'/(2\log(n))$ , i.e.,  $S^{**} = \{|\{i \in [s'] : 2i + \sigma_i \in B\}| < s^{**}\}$ .

**Lemma 4.**  *$S^{**}$  is a high probability event, i.e.  $\Pr_{\mu'}(S^{**}) = \Pr_{\mu}(S^{**}) \geq 1 - \exp(-s'c^*/(13\log n))$ . This is greater than  $1 - \epsilon$  for sufficiently large  $n$ .*

Let  $\nu$  be the distribution  $\mu'$  conditioned on the events  $S^{**}$  and  $E^*$ . Alice and Bob can easily determine the prefix and suffix of a stream according to this distribution:

1. Alice randomly places the special items such that at most  $c^*s'/(2\log n)$  distinct elements occur in the suffix, and chooses a value  $t$  with probability  $\Pr_{\mu'}(E_t|S^{**})/(1 - \Pr_{\mu'}(E^*|S^{**}))$ . She then randomly places  $(n_1 - s_1 - t)/2$  “0”’s and  $(n_1 - s_1 + t)/2$  “2s+2”’s and the special items she assigned to the suffix. She then sends  $S' = \{(i, \sigma_i) : 2i + \sigma_i \notin \text{prefix of stream}\}$  (note that this is a multi-set in general) to Bob along with the value of  $t$ .
2. Bob randomly places  $x - (n_1 - s - t)/2$  “0”’s and  $y - (n_1 - s + t)/2$  “2s+2”’s and  $\{2i + \sigma_i : (i, \sigma_i) \in S'\}$  in the suffix of the stream.

To prove our result we need to show that  $\nu$  is sufficiently close to  $\mu$ . This can be shown by appealing to Lemmas 3 and 4.

**Lemma 5.**  *$\nu$  is  $5\epsilon$ -near to random, i.e.,  $\|\mu - \nu\|_1 \leq 5\epsilon$ .*

**Theorem 2.** *Computing an  $n^\delta$ -approximate median of a random order stream with probability at least  $9/10$  requires  $\Omega(\sqrt{n^{1-3\delta}/\log(n)})$  space.*

*Proof.* Let  $\mathcal{A}$  be an algorithm using  $M$  bits of memory that returns the median of a randomly ordered stream with probability  $9/10$  (over both the ordering and the private coin tosses). Assume Alice and Bob generate the stream as described above. In addition, Alice runs  $\mathcal{A}$  on the prefix of the stream and sends the memory state to Bob when she is done. Bob then continues running  $\mathcal{A}$ , initialized with the transmitted memory state, on the suffix of the stream. Bob then returns 1 if the output of the algorithm is odd and 0 otherwise. By Lemma 1 and Lemma 5 this protocol is correct with probability  $9/10 - 5\epsilon$ .

We now bound the number of bits required to follow the above protocol. The number of bits required to specify a sub-set of the unique elements of  $S$  of size at most  $s^{**}$  is

$$\lg \sum_{0 \leq s_2 \leq s^{**}} \binom{s'}{s_2} \leq \lg(s^{**} + 1) + s^{**} \lg \left( \frac{s'e}{s^{**}} \right) .$$

For each unique element occurring in the suffix of the stream we need to specify how many time it occurs and the associated bit value of  $\sigma$ . This takes at most  $s^{**}(1 + \lg n)$  bits. Hence the number of bits transmitted in the protocol is at most

$$\lg s^{**} + s^{**} \lg \left( \frac{s'e}{s^{**}} \right) + s^{**}(1 + \lg n) + \lg n + M,$$

Assuming that  $s^{**} = \omega(\lg n)$  this is bounded above by  $\frac{3}{4}c^*s + M$  and hence we conclude that  $M = \Omega(s')$  by appealing to Theorem 1.

### 4 Adversarial Order Lower-Bound

In this section we will prove that any  $k$  pass algorithm that returns the median of an adversarially ordered stream must use  $\tilde{\Omega}(n^{1/k})$  space. This, coupled with the upper bound of Munro and Paterson [21], will resolve the space complexity of multi-pass algorithms for median finding up to poly-logarithmic terms. The proof will use a reduction from the communication complexity of a generalized form of pointer chasing that we now describe.

**Definition 3 (Generalized Pointer Chasing).** For  $i \in [k]$ , let  $f_i : [m] \rightarrow [m]$  be an arbitrary function. Then  $g_k$  is defined by

$$g_k(f_1, f_2, \dots, f_k) = f_k(f_{k-1}(\dots f_1(1)\dots)) .$$

Let the  $i$ -th player,  $P_i$ , have function  $f_i$  and consider a protocol in which the players must speak in the reverse order, i.e.,  $P_k, P_{k-1}, \dots, P_1, P_k, \dots$ . We say the protocol has  $r$  rounds if  $P_k$  speaks  $r$  times. Let  $R_\delta^r(g_k)$  be the total number of bits that must be communicated in an  $r$ -round (randomized) protocol for  $P_1$  to learn  $g_k$  with probability at least  $1 - \delta$ .

Note that  $R_0^k(g_k) = O(k \log m)$ . We will be looking at  $k$  round protocols. The proof of the next result follows along similar lines to [22] and will be proved in the Appendix B.

**Theorem 3.**  $R_{1/10}^{k-1}(g_k) = \Omega(m/k^4 - k^2 \log m)$ .

The next theorem is shown by reducing generalized pointer-chasing to approximate selection.

**Theorem 4.** Finding an  $n^\delta$ -approximate median in  $k$  passes of an adversarially ordered stream requires  $\Omega(n^{\frac{1-\delta}{k}} k^{-6})$ .

*Proof.* We will show how a  $k$ -pass algorithm  $\mathcal{A}$  that computes a  $t$ -approximate median of a length  $n$  stream gives rise to a  $k$ -round protocol for computing  $g_{k+1}$  when  $m = (n/((k+1)(2t+1)))^{1/k} / 2$ . If  $\mathcal{A}$  uses  $M$  bits of space then the protocol uses at most  $(k(k+1) - 1)M$  bits. Hence by Theorem 3, this implies that  $M = \Omega(m/k^6) = \Omega((n/t)^{1/k} k^{-6})$ .

The intuition behind the proof is that any  $t$ -approximate median will correspond to a number  $g_1 g_2 g_3 \dots g_{k+1}$  written in base  $m + 2$ . The input of  $P_1$  will first determine the highest order ‘bit’, i.e.,  $g_1$ . Then the input of  $P_2$  will determine the  $g_2$  and so on. Specifically, each player  $P_i$  will determine a segment of the stream  $S_i$ :  $P_{k+1}$  determines the first  $n_{k+1} = |S_{k+1}|$  elements,  $P_k$  determines the next  $n_k = |S_k|$ , etc. These segments are defined as follows,

$$S_1 = \left\{ \underbrace{0, \dots, \dots, 0}_{(m-f_1(1))(2t+1)(2m-1)^{k-1}}, \underbrace{(m+1)b^k, \dots, (m+1)b^k}_{(f_1(1)-1)(2t+1)(2m-1)^{k-1}} \right\}$$

**Table 1.** Reduction from Pointer Chasing to Exact Median Finding. A triple of the form  $(x_2, x_1, x_0)$  corresponds to the numerical value  $x_2 \cdot 5^2 + x_1 \cdot 5^1 + x_0 \cdot 5^0$ . Note that  $\text{median}(S_1 \cup S_2 \cup S_3) = f_A(1) \cdot 5^2 + f_B(f_A(1)) \cdot 5^1 + f_C(f_B(f_A(1))) \cdot 5^0$ .

| $S_1$                            | $S_2$  | $S_3$  |
|----------------------------------|--|--|
| $(0, 0, 0) \times 5(3 - f_A(1))$ | $(1, 0, 0) \times (3 - f_B(1))$                  | $(1, 1, f_C(1)), (1, 2, f_C(2)), (1, 3, f_C(3))$ |
| $(2, 0, 0) \times (3 - f_B(2))$  | $(2, 1, f_C(1)), (2, 2, f_C(2)), (2, 3, f_C(3))$ | $(2, 1, f_C(1)), (2, 2, f_C(2)), (2, 3, f_C(3))$ |
| $(3, 0, 0) \times (3 - f_B(3))$  | $(3, 1, f_C(1)), (3, 2, f_C(2)), (3, 3, f_C(3))$ | $(3, 1, f_C(1)), (3, 2, f_C(2)), (3, 3, f_C(3))$ |
| $(4, 0, 0) \times 5(f_A(1) - 1)$ | $(4, 4, 0) \times (f_B(3) - 1)$                  |  |

and for  $j \in \{2, \dots, k\}$ ,

$$S_j = \bigcup_{x_{k+2-j}, \dots, x_k \in [m]} \left\{ \underbrace{\sum_{i=k+2-j}^k x_i b^i, \dots, \sum_{i=k+2-j}^k x_i b^i}_{(m - f_j(x_{k+2-j}))(2t+1)(2m-1)^{k-j}}, \right. \\ \left. \underbrace{(m+1)b^{k+1-j} + \sum_{i=k+2-j}^k x_i b^i, \dots, (m+1)b^{k+2-j} + \sum_{i=k+2-j}^k x_i b^i}_{(f_j(x_{k+2-j})-1)(2t+1)(2m-1)^{k-j}} \right\},$$

and finally,

$$S_{k+1} = \bigcup_{x_1, \dots, x_k \in [m]} \left\{ \underbrace{f_{k+1}(x_1) + \sum_{i=1}^k x_i b^i, \dots, f_{k+1}(x_1) + \sum_{i=1}^k x_i b^i}_{2t+1} \right\},$$

where  $b = m + 2$ . See Table [II](#) for the an example when  $k = 2$  and  $m = 3$ . Note that  $n_{k+1} = (2t + 1)m^k$  and for  $j \geq k$ ,  $n_j = (2t + 1)(m - 1)(2m - 1)^{k-j+1}m^{j-1} < (2t + 1)m^k$ . Hence,  $\sum_{j \in [k+1]} n_j \leq (2t + 1)(k + 1)(2m)^k = n$ , and that the largest value in the stream is  $(m + 1)b^k = O(n)$ . Note that any  $t$ -approximate median equals,  $\sum_{i \in [k+1]} g_i b^{k+1-i}$  and thus if  $P_1$  returns the  $t$ -approximate median modulo  $b$  then this is  $g_{k+1}$ . This can easily be computed by a protocol in which each player transmits the memory state of the algorithm at the appropriate juncture.



## References

1. Chan, T.M., Chen, E.Y.: Multi-pass geometric algorithms. In: Symposium on Computational Geometry, pp. 180–189 (2005)
2. Chang, K.L., Kannan, R.: The space complexity of pass-efficient algorithms for clustering. In: ACM-SIAM Symposium on Discrete Algorithms, pp. 1157–1166 (2006)
3. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In: PODS, pp. 263–272 (2006)
4. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. *J. Algorithms* 55(1), 58–75 (2005)
5. Demaine, E.D., López-Ortiz, A., Munro, J.I.: Frequency estimation of internet packet streams with limited space. In: Möhring, R.H., Raman, R. (eds.) *ESA 2002*. LNCS, vol. 2461, pp. 348–360. Springer, Heidelberg (2002)
6. Drineas, P., Kannan, R.: Pass efficient algorithms for approximating large matrices. In: Symposium, A.C.M.-S.I.A.M. (ed.) *ACM-SIAM Symposium on Discrete Algorithms*, pp. 223–232 (2003)
7. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. *Theoretical Computer Science* 348(2-3), 207–216 (2005)
8. Feigenbaum, J., Kannan, S., Strauss, M., Viswanathan, M.: Testing and spot-checking of data streams. *Algorithmica* 34(1), 67–80 (2002)
9. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: How to summarize the universe: Dynamic maintenance of quantiles. In: *International Conference on Very Large Data Bases (VLDB)*, pp. 454–465 (2002)
10. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. In: *ACM SIGMOD International Conference on Management of Data*, pp. 58–66. ACM Press, New York (2001)
11. Guha, S., McGregor, A.: Approximate quantiles and the order of the stream. In: *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 273–279. ACM Press, New York (2006)
12. Guha, S., McGregor, A.: Space-efficient sampling. In: *AISTATS*, pp. 169–176 (2007)
13. Guha, S., McGregor, A., Venkatasubramanian, S.: Streaming and sublinear approximation of entropy and information distances. In: *ACM-SIAM Symposium on Discrete Algorithms*, pp. 733–742 (2006)
14. Gupta, A., Zane, F.: Counting inversions in lists. In: *ACM-SIAM Symposium on Discrete Algorithms*, pp. 253–254 (2003)
15. Kannan, S.: Open problems in streaming. *DIMACS Workshop* (2001), (Slides: [dimacs.rutgers.edu/Workshops/Streaming/abstracts.html](http://dimacs.rutgers.edu/Workshops/Streaming/abstracts.html))
16. Kushilevitz, E., Nisan, N.: *Communication Complexity*. Cambridge University Press, Cambridge (1997)
17. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: Approximate medians and other quantiles in one pass and with limited memory. In: *ACM SIGMOD International Conference on Management of Data*, pp. 426–435. ACM Press, New York (1998)
18. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: Random sampling techniques for space efficient online computation of order statistics of large datasets. In: *ACM SIGMOD International Conference on Management of Data*, pp. 251–262. ACM Press, New York (1999)
19. McGregor, A.: Finding graph matchings in data streams. In: *APPROX-RANDOM*, pp. 170–181 (2005)

20. Miltersen, P.B., Nisan, N., Safra, S., Wigderson, A.: On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.* 57(1), 37–49 (1998)
21. Munro, J.I., Paterson, M.: Selection and sorting with limited storage. *Theor. Comput. Sci.* 12, 315–323 (1980)
22. Nisan, N., Wigderson, A.: Rounds in communication complexity revisited. *SIAM J. Comput.* 22(1), 211–219 (1993)
23. Shrivastava, N., Buragohain, C., Agrawal, D., Suri, S.: Medians and beyond: new aggregation techniques for sensor networks. In: *SenSys*, pp. 239–249 (2004)
24. Yao, A.C.: Lower bounds by probabilistic arguments. In: *IEEE Symposium on Foundations of Computer Science*, pp. 420–428. IEEE Computer Society Press, Los Alamitos (1980)

## A Selection Algorithm for Random-Order Streams

In this section we show how to perform approximate selection in a single pass. We will also assume that the stream contains distinct values. This can easily be achieved by attaching a secondary value  $y_i \in_R [n^3]$  to each item  $x_i$  in the stream. We say  $(x_i, y_i) < (x_j, y_j)$  iff  $x_i < x_j$  or  $(x_i = x_j \text{ and } y_i < y_j)$ . Note that breaking the ties arbitrarily results in a stream whose order is not random.

Our algorithm proceeds in phases and each phase is composed of three distinct sub-phases; the *Sample* sub-phase, the *Estimate* sub-phase, and the *Update* sub-phase. At all points we maintain an open interval  $(a, b)$  such that we believe that the value of the element with rank  $k$  is between  $a$  and  $b$ . In each phase we aim to narrow the interval  $(a, b)$ . The *Sample* sub-phase finds a value  $u \in (a, b)$ . The *Estimate* sub-phase estimates the rank of  $u$ . The *Update* sub-phase replaces  $a$  or  $b$  by  $u$  depending on whether the rank of  $u$  is believed to be less or greater than  $u$ . The algorithm is presented in Fig. 1.

**Theorem 5.** *When presented with a randomly ordered stream, the Selection algorithm returns a value  $u \in S$  such that  $\text{RANK}_S(u) = k \pm 10 \ln^2(n) \ln(\delta^{-1}) \sqrt{k}$  with probability at least  $1 - \delta$ . The algorithm uses only  $O(\log n)$  bits of space.*

The algorithm appears to need to know the length of the stream in advance but this assumption can be removed by making multiple “staggered” instantiations of the algorithm that correspond to guesses of the length as in [11]. Also, as with the algorithm presented in [11], the algorithm can be used as a sub-routine to create an algorithm that performs exact selection in  $O(\log \log n)$  passes. Lastly, the algorithm can be generalized to deal with streams whose order is only “almost random” in the sense of being  $\epsilon$ -generated random or  $t$ -bounded random [11].

## B Proof of Theorem 3

The proof is a generalization of a proof by Nisan and Wigderson [22]. We present the entire argument for completeness. In the proof we lower bound the  $(k - 1)$ -round distributional complexity,  $D_{1/20}^{k-1}(g_k)$ , i.e. we will consider a deterministic

**Selection Algorithm:**

1. Let  $\Upsilon = 10 \ln^2(n) \ln(\delta^{-1})\sqrt{k}$  and  $p = 2(\lg(n/\Upsilon) + \sqrt{\ln(3/\delta) \lg(n/\Upsilon)})$
2. Let  $a = -\infty$  and  $b = +\infty$
3. Let  $l_1 = n\Upsilon^{-1} \ln(3n^2p/\delta)$  and  $l_2 = 2(n-1)\Upsilon^{-1} \sqrt{(k+\Upsilon) \ln(6np/\delta)}$ .
4. Let the stream  $S = S_1, E_1, \dots, S_p, E_p$  where  $|S_i| = l_1$  and  $|E_i| = l_2$
5. Phase  $i$ :
  - (a) *Sample* sub-phase: If  $S_i \cap (a, b) = \emptyset$  return  $a$ , else let  $u \in S_i \cap [a, b]$
  - (b) *Estimate* sub-phase: Compute  $r = \text{RANK}_{E_i}(u)$  and  $\tilde{r} = \frac{(n-1)(r-1)}{l_2} + 1$ .
  - (c) *Update* sub-phase: If  $\tilde{r} < k - \Upsilon/2$ ,  $a \leftarrow u$ ,  $\tilde{r} > k + \Upsilon/2$ ,  $b \leftarrow u$  else return  $u$

**Fig. 1.** An Algorithm for Computing Approximate Quantiles

protocol and an input chosen from some distribution. The theorem will then follow by Yao’s Lemma [24] since  $D_{1/20}^{k-1}(g_k) \leq 2R_{1/10}^{k-1}(g_k)$ .

Let  $T$  be the protocol tree of a deterministic  $k$ -round protocol. We consider the input distribution where each  $f_i$  is chosen uniformly and independently from  $F$ , the set of all  $m^m$  functions from  $[m]$  to  $[m]$ . Note that this distribution over inputs gives rise to distribution over paths from the root of  $T$  to the leaves. We will assume that in round  $j$ ,  $P_i$ ’s message includes  $g_{j-1}$  if  $i > j$  and  $g_j$  if  $i \leq j$ .

By induction this is possible with only  $O(k^2 \log m)$  extra communication. Consequently we may assume that at each node at least  $\lg m$  bits are transmitted. We will assume that protocol  $T$  requires at most  $\epsilon m/2$  bits of communication where  $\epsilon = 10^{-4}(k+1)^{-4}$  and derive a contradiction.

Consider a node  $z$  in the protocol tree of  $T$  corresponding to the  $j$ th round of the protocol when it is  $P_i$ ’s turn to speak. Let  $g_{t-1}$  be the appended information in the last transmission. Note that  $g_0, g_1, \dots, g_{t-1}$  are specified by the messages so far.

Denote the set of functions  $f_1 \times \dots \times f_k$  that are consistent with the messages already sent be  $F_1^z \times \dots \times F_k^z$ . Note that the probability of arriving at node  $z$  is  $|F|^{-k} \prod_{1 \leq j \leq k} |F_j^z|$ . Also note that, conditioned on arriving at node  $z$ ,  $f_1 \times \dots \times f_k$  is uniformly distributed over  $F_1^z \times \dots \times F_k^z$ .

Let  $c_z$  be the total communication until  $z$  is reached. We say a node  $z$  in the protocol tree is *nice* if, for  $\delta = \max\{4\sqrt{\epsilon}, 400\epsilon\}$ , it satisfies the following two conditions:

$$|F_j^z| \geq 2^{-2c_z} |F| \text{ for } j \in [k] \quad \text{and} \quad H(f_t^z(g_{t-1})) \geq \lg m - \delta .$$

*Claim.* Given the protocol reaches node  $z$  and  $z$  is nice then,

$$\Pr[\text{next node visited is nice}] \geq 1 - 4\sqrt{\epsilon} - 1/m .$$

*Proof.* Let  $w$  be a child of  $z$  and let  $c_w = c_z + a_w$ . For  $l \neq i$  note that  $|F_l^w| = |F_l^z|$  since  $P_l$  did not communicate at node  $z$ . Hence the probability that we reach node  $w$  given we have reached  $z$  is  $\prod_{1 \leq j \leq k} |F_j^w|/|F_j^z| = |F_i^w|/|F_i^z|$ . Furthermore, since  $z$  is nice,

$$\Pr [|F_i^w| < 2^{-2c_w} |F|] \leq \Pr \left[ \frac{|F_i^w|}{|F_i^z|} < 2^{-2a_w} \right] \leq \sum_w 2^{-2a_w} \leq \frac{1}{m} \sum_w 2^{-a_w} \leq \frac{1}{m} .$$

where the second last inequality follows from  $a_w \geq \lg m$  and the last inequality follows by Kraft’s inequality (the messages sent must be prefix free.) Hence, with probability at least  $1 - 1/m$ , the next node in the protocol tree satisfies the first condition for being nice.

Proving the second condition is satisfied with high probability is more complicated. Consider two different cases,  $i \neq t$  and  $i = t$ , corresponding to whether or not player  $i$  appended  $g_t$ . In the first case, since  $P_t$  did not communicate,  $F_t^z = F_t^w$  and hence  $H(f_t^w(g_{t-1})) = H(f_t^z(g_{t-1})) \geq \lg m - \delta$ .

We now consider the second case. We need to show that  $H(f_{t+1}^w(g_t)) \geq \lg m - \delta$ . Note that we can express  $f_{t+1}^w$  as the following vector of random variables,  $(f_{t+1}^w(1), \dots, f_{t+1}^w(m))$  where each  $f_{t+1}^w(v)$  is a random variables in universe  $[m]$ . Note there is no reason to believe that components of this vector are independent. By the sub-additivity of entropy,

$$\sum_{v \in [m]} H(f_{t+1}^w(v)) \geq H(f_{t+1}^w) \geq \lg(2^{-2c_w} |F|) = \lg(|F|) - 2c_w \geq m \lg m - \epsilon m$$

using the fact that  $f_{t+1}^w$  is uniformly distribution over  $F_{t+1}^w$ ,  $|F_{t+1}^w| \geq 2^{-2c_w} |F|$  and  $c_w \leq \epsilon m/2$ . Hence if  $v$  were chosen uniformly at random from  $[m]$ ,

$$\Pr [H(f_{t+1}^w(v)) \leq \log m - \delta] \leq \epsilon/\delta ,$$

by Markov’s inequality. However, we are not interested in a  $v$  chosen uniformly at random but rather  $v = g_t = f_t^z(g_{t-1})$ . However since the entropy of  $f_t^z(g_{t-1})$  is large it is almost distributed uniformly. Specifically, since  $H(f_t^z(g_{t-1})) \geq \lg m - \delta$  it is possible to show that (see [22]), for our choice of  $\delta$ ,

$$\Pr [H(f_{t+1}^w(g_t)) \leq \log m - \delta] \leq \frac{\epsilon}{\delta} \left( 1 + \sqrt{\frac{4\delta}{\epsilon/\delta}} \right) \leq 4\sqrt{\epsilon} .$$

Hence with probability at least  $1 - 4\sqrt{\epsilon}$  the next node satisfies the second condition of being nice. The claim follows by the union bound.

Note that the height of the protocol tree is  $k(k - 1)$  and that the root of the protocol tree is nice. Hence the probability of ending at a leaf that is not nice is at most  $k(k - 1)(1/m + 4\sqrt{\epsilon}) \leq 1/25$ . If the final leaf node is nice then then  $H(g_t)$  is at least  $\lg m - \delta$  and hence the probability that  $g_t$  is guessed correctly is at most  $(\delta + 1)/\lg m$  using Fano’s inequality. This is less than  $1/100$  for sufficiently large  $m$  and hence the total probability of  $P_1$  guessing  $g_k$  correctly is at most  $1 - 1/20$ .

# Streaming and Fully Dynamic Centralized Algorithms for Constructing and Maintaining Sparse Spanners

Michael Elkin

<sup>1</sup> Department of Computer Science, Ben-Gurion University of the Negev,  
Beer-Sheva, Israel

elkinm@cs.bgu.ac.il

<sup>2</sup> This research has been supported by the Israeli Academy of Science, grant 483/06

**Abstract.** We present a streaming algorithm for constructing sparse spanners and show that our algorithm out-performs significantly the state-of-the-art algorithm for this task [20]. Specifically, the *processing time-per-edge* of our algorithm is drastically smaller than that of the algorithm of [20], and all other efficiency parameters of our algorithm are no greater (and some of them are strictly smaller) than the respective parameters for the state-of-the-art algorithm.

We also devise a fully dynamic centralized algorithm maintaining sparse spanners. This algorithm has a very small incremental update time, and a non-trivial decremental update time. To our knowledge, this is the first fully dynamic centralized algorithm for maintaining sparse spanners that provides non-trivial bounds on both incremental and decremental update time for a wide range of stretch parameter  $t$ .

## 1 Introduction

The study of the streaming model became an important research area after the seminal papers of Alon, Matias and Szegedy [1], and Feigenbaum et al. [21] were published. More recently, research in the streaming model was extended to traditional graph problems [8,19,18,20]. The input to a graph algorithm in the streaming model is a sequence (or *stream*) of edges representing the edge set  $E$  of the graph. This sequence can be an arbitrary permutation of the edge set  $E$ .

In this paper we devise a streaming algorithm for constructing sparse spanners for unweighted undirected graphs. Informally, graph *spanners* can be thought of as sparse skeletons of communication networks that approximate to a significant extent the metric properties of the respective networks. Spanners serve as an underlying graph-theoretic construct for a great variety of distributed algorithms. Their most prominent applications include *approximate distance computation* [15,18,14], *synchronization* [4,22,6], *routing* [23,7,26], and online load balancing [5]. The problem of constructing spanners with various parameters is a subject of intensive recent research [17,15,25,12,11,27,24,28].

The state-of-the-art streaming algorithm for computing a sparse spanner for an input (unweighted undirected)  $n$ -vertex graph  $G = (V, E)$  was presented in a

recent breakthrough paper of Feigenbaum et al. [20]. For an integer parameter  $t \geq 2$ , their algorithm, with high probability, constructs a  $(2t - 1)$ -spanner with  $O(t \cdot \log n \cdot n^{1+1/(t-1)})$  edges *in one pass* over the input using  $O(t \cdot \log^2 n \cdot n^{1+1/(t-1)})$  bits of space. It processes each edge in the stream in time  $O(t^2 \cdot \log n \cdot n^{1/(t-1)})$ . Their result also immediately gives rise to a streaming algorithm for  $(2t - 1)$ -approximate all-pairs-distance-computation (henceforth,  $(2t - 1)$ -APDC) algorithm with the same parameters.

Our algorithm constructs a  $(2t - 1)$ -spanner with  $O((t \cdot \log n)^{1-1/t} \cdot n^{1+1/t})$  edges in one pass over the input using  $O(t^{1-1/t} \cdot \log^{2-1/t} n \cdot n^{1+1/t})$  bits of space, for an integer parameter  $t \geq 1$ . (The size of the spanner and the number of bits are with high probability.) Most importantly, the *processing time-per-edge* of our algorithm is *drastically smaller* than that of Feigenbaum et al. [20]. Specifically, the expectation of the processing time-per-edge in our algorithm is  $O(1)$ , it is  $O\left(\sqrt{\frac{\log \log n}{\log^{(3)} n}}\right)$  with high probability, and in the worst-case processing an edge  $e = (v, u)$  requires  $O\left(\sqrt{\frac{\log \deg(e)}{\log \log \deg(e)}}\right)$ , where  $\deg(e) = \max\{\deg(v), \deg(u)\}$ .

To summarize, our algorithm constructs a spanner with a smaller number of edges and number of bits of space used (by a factor of  $n^{1/(t(t-1))}(t \cdot \log n)^{1/t}$ ), and it does so using a *drastically* reduced (and very close to optimal) *processing time-per-edge, at no price whatsoever*. Our result also gives rise to an improved streaming  $(2t - 1)$ -approximate APDC algorithm with the same parameters. Observe also that for the stretch guarantee equal to 3, our algorithm is the first one to provide a non-trivial bound. A concise comparison of our result with the state-of-the-art result of Feigenbaum et al. [20] can be found in Table 1.

Independently of us Baswana [9] came up with a streaming algorithm for computing sparse spanners. The efficiency parameters of the algorithm of [9] are equal to those of our algorithm, except that it provides *amortized* bound of  $O(1)$  on the processing time-per-edge. However, processing an individual edge by this algorithm may require  $\Omega(n)$  time in the worst-case, and [9] provides no guarantee on the expectation of the processing time-per-edge. Also, the algorithm of [9] appears to be significantly more complex than ours.

A variant of our algorithm can be seen as a fully dynamic centralized algorithm for maintaining a  $(2t - 1)$ -spanner with  $O((t \cdot \log n)^{1-1/t} n^{1+1/t})$  edges. The incremental update time of this algorithm is exactly the processing time-per-edge of our streaming algorithm, and, in fact, the way that the dynamic algorithm processes incremental updates is identical to the way that our streaming algorithm processes each edge of the stream. The expected decremental update time (the time required to update the data structures of the algorithm when an edge  $e$  is deleted) is  $O\left(\frac{n}{n^{1/t}} \cdot (t \log n)^{1/t}\right)$ , and moreover, with probability at least  $1 - \left(\frac{t \cdot \log n}{n}\right)^{1/t}$ , the decremental update time is  $O\left(\sqrt{\frac{h}{\log h}}\right)$ , where  $h = \max\{\log \deg(e), \log \log n\}$ . The size of the data structures maintained by an incremental variant of our algorithm is  $O(t^{1-1/t} \cdot (\log n)^{2-1/t} \cdot n^{1+1/t})$  bits. To cope with decremental updates as well, our algorithm needs to maintain  $O(|E| \cdot \log n)$  bits. (Note that the incremental algorithm maintains data

**Table 1.** A comparison between the algorithm of Feigenbaum et al. [20] and our new streaming algorithm. The degree of an edge  $e = (u, v)$  is  $\max\{\text{deg}(v), \text{deg}(u)\}$ . The word “whp” stands for “with high probability”. The result of [20] applies for  $t \geq 2$ , while our algorithm applies for  $t \geq 1$ .

|      | # passes | Processing time-per-edge   | Stretch  | # Edges (whp)                                 |
|------|----------|--|----------|---|
| [20] | 1        | whp $O(t^2 \cdot \log n \cdot n^{1/(t-1)})$  | $2t - 1$ | $O(t \cdot \log n \cdot n^{1+1/(t-1)})$       |
| New  | 1        | Expected $O(1)$ ,<br>whp $O\left(\sqrt{\frac{\log \log n}{\log^{(3)} n}}\right)$ ,<br>worst-case $O\left(\sqrt{\frac{\log \text{deg}(e)}{\log \log \text{deg}(e)}}\right)$ | $2t - 1$ | $O((t \cdot \log n)^{1-1/t} \cdot n^{1+1/t})$ |

structures of overall size *sublinear* in the size of the input  $|E|$ . This is not really surprising, since this is essentially a streaming algorithm.)

To our knowledge, this is the first fully dynamic centralized algorithm for maintaining sparse spanners that provides non-trivial bounds for a wide range of stretch parameter  $t$ . The first algorithm of this type was devised recently by Ausillo et al. [3]. This algorithm maintains 3- and 5-spanners of optimal size with  $O(n)$  amortized time per operation for an intermixed sequence of  $\Omega(n)$  edge insertions and deletions.

Very recently Baswana [10] improved the result of Ausillo et al. [3] and devised a fully dynamic algorithm for maintaining spanners of optimal size with stretch at most 6 with expected constant update time. Baswana [10] presented also a *decremental* algorithm for maintaining  $(2t - 1)$ -spanner of optimal size with expected update time of  $O(t^2 \cdot \log^2 n)$ . However, the latter algorithm provides no non-trivial bound for *incremental* update time. Also, note that with high probability the decremental update time of our algorithm is significantly smaller than that of [10].

**Related Work and Our Techniques:** A fully dynamic distributed algorithm for maintaining sparse spanners is presented in the companion paper [16]. Our algorithm in this paper combines the techniques of Feigenbaum et al. [20], of Baswana and Sen [12], with those developed in [16].

More specifically, both the algorithm of Feigenbaum et al. [20] and our algorithm build upon the techniques of Baswana and Sen [12]. Both algorithms label vertices by numbers, and use the labels to decide whether a newcoming edge needs to be inserted into the spanner or not. The main conceptual difference between the two algorithms is that in the algorithm of Feigenbaum et al. [20] for every vertex  $v$ , the entire list of labels  $L$  such that  $v$  was ever labeled by  $L$  is stored. These lists are then manipulated in a rather sophisticated manner to ensure that only the “right” edges end up in the spanner. On the other hand, in our algorithm only one (current) label is stored for every vertex, and decisions are made on the basis of this far more restricted information. As a result, our

algorithm avoids manipulating lists of labels, and is, consequently, much simpler, and far more efficient.

One could expect that using a smaller amount of information to make decisions may result in a denser spanner, or/and in a spanner with a relaxed stretch guarantee. Surprisingly, however, we show that this is not the case, and that the parameters of spanners produced by our algorithm are better than the parameters of spanners produced by the algorithm of Feigenbaum et al. [20].

**Preliminaries:** For a parameter  $\alpha$ ,  $\alpha \geq 1$ , a subgraph  $G'$  of the graph  $G = (V, E)$  is called an  $\alpha$ -spanner of  $G$  if for every pair of vertices  $x, y \in V$ ,  $dist_{G'}(x, y) \leq \alpha \cdot dist_G(x, y)$ , where  $dist_G(u, w)$  denotes the distance between  $u$  and  $w$  in  $G$ . The parameter  $\alpha$  is called the *stretch* or *distortion* parameter of the spanner. Also, for a fixed value of  $t = 1, 2, \dots$ , we say that a subgraph  $G'$  spans an edge  $e = (v, u) \in E$ , if  $dist_{G'}(v, u) \leq 2t - 1$ .

**Structure of this paper:** In Section 2 we present and analyze our streaming algorithm. In Section 3 we extend it to the dynamic centralized setting. Most proofs are omitted from this extended abstract.

## 2 The Streaming Model

In this section we present and analyze the version of our algorithm that constructs spanners in the streaming model of computation.

### 2.1 The Algorithm

The algorithm accepts as input a *stream* of edges of the input graph  $G = (V, E)$ , and an integer positive parameter  $t$ , and constructs a  $(2t - 1)$ -spanner  $G' = (V, H)$ ,  $H \subseteq E$ , of  $G$  with  $O((t \cdot \log n)^{1-1/t} \cdot n^{1+1/t})$  edges using only  $O(|H| \cdot \log n) = O(t^{1-1/t}(\log n)^{2-1/t} \cdot n^{1+1/t})$  bits of storage space, and processing each edge in  $O(1)$  expected time, in one pass over the stream. Note that the space used by the algorithm is linear in the size of the representation of the spanner. Regarding the processing time-per-edge, processing the edge  $e$  requires  $O\left(\sqrt{\frac{\log deg(e)}{\log \log deg(e)}}\right)$  time in the worst-case, and moreover, with high probability, the processing time-per-edge is  $O\left(\sqrt{\frac{\log \log n}{\log^{(3)} n}}\right)$ .

At the beginning of the execution (before the first edge of the stream arrives), the vertices of  $V$  are assigned unique identifiers from the set  $\{1, 2, \dots, n\} = [n]$ ,  $n = |V|$ . (Henceforth, for any positive integer  $k$ , the set  $\{1, 2, \dots, k\}$  is denoted  $[k]$ , and the set  $\{0, 1, \dots, k\}$  is denoted  $[(k)]$ .) Let  $I(v)$  denote the identifier of the vertex  $v$ . Also, as a part of preprocessing, the algorithm picks a non-negative integer *radius*  $r(v)$  for every vertex  $v$  of the graph from the *truncated geometric probability distribution* given by  $\mathbb{P}(r = k) = p^k \cdot (1 - p)$ , for every  $k \in [(t - 2)]$ , and  $\mathbb{P}(r = t - 1) = p^{t-1}$ , with  $p = \left(\frac{t \log n}{n}\right)^{1/t}$ . Note that this distribution satisfies  $\mathbb{P}(r \geq k + 1 \mid r \geq k) = p$  for every  $k \in [(t - 2)]$ .



We next introduce a few definitions that will be useful for the description of our algorithm. During the execution, the algorithm maintains for every vertex  $v$  the variable  $P(v)$ , called the *label* of  $v$ , initialized as  $I(v)$ . The labels of vertices may grow as the execution proceeds, and they accept values from the set  $\{1, 2, \dots, n \cdot t\}$ . A label  $P$  in the range  $i \cdot n + 1 \leq P < (i + 1)n$ , for  $i \in [(t - 1)]$  is said to be a label of *level*  $i$ ; in this case we write  $L(P) = i$ . The value  $B(P)$  is given by  $B(P) = n$  if  $n$  divides  $P(v)$ , and by  $B(P) = P(v) \pmod{n}$ , otherwise. This value is called the *base value* of the label  $P$ . The vertex  $w = w_P$  such that  $I(w) = B(P)$  is called the *base vertex* of the label  $P$ . A label  $P$  is said to exist if the level  $L(P)$  of  $P$  is no greater than the radius of the base vertex  $w_P$ , i.e.,  $L(P) \leq r(w_P)$ . The label  $P$  is called *selected* if  $L(P) < r(w_P)$ . Note that for a label  $P$  to be selected, it must satisfy  $L(P) \leq t - 2$ .

One of the basic primitives of the algorithm is comparing the labels. We say that the labels  $P(v)$  and  $P(v')$  of the vertices  $v$  and  $v'$ , respectively, satisfy the relation  $P(v) \succ P(v')$  if and only if either  $P(v) > P(v')$  or  $(P(v) = P(v')$  and  $I(v) > I(v'))$ . Note that for every two vertices  $v$  and  $v'$ , either  $P(v) \succ P(v')$  or  $P(v') \succ P(v)$ .

For a label  $P$  of level  $t - 2$  or smaller,

$$\mathbb{P}(P \text{ is selected}) = \mathbb{P}(r(w_P) \geq L(P) + 1 \mid r(w_P) \geq L(P)) = p.$$

**Lemma 1.** *With high probability, the number of distinct labels of level  $t - 1$  that occur in the algorithm is  $O(n^{1/t} \cdot (t \cdot \log n)^{1-1/t})$ .*

We remark that the way that we define and manipulate labels is closely related to the way it is done in Feigenbaum et al. [20].

For every vertex the algorithm maintains an edge set  $Sp(v)$ , initialized as an empty set. During the execution the algorithm inserts some edges into  $Sp(v)$ , and never removes them. In other words, the sets  $Sp(v)$  grow monotonely during the execution of the algorithm. It is useful to think of the sets  $Sp(v)$  as divided into two disjoint subsets  $T(v)$  and  $X(v)$ ,  $Sp(v) = T(v) \cup X(v)$ . The set  $T(v)$  is called the set of the *tree edges* of the vertex  $v$ , and the set  $X(v)$  is called the set of the *cross edges* of the vertex  $v$ . During the execution the algorithm constructs implicitly a *tree cover* of the graph. The edges of this tree cover are (implicitly) maintained in the sets  $T(v)$ . In addition, the spanner will also contain some edges that connect different trees of the tree cover; these edges are (implicitly) maintained in the sets  $X(v)$ . Each edge  $e$  that is (implicitly) inserted into the set  $T(v)$  will also be labeled by a label of  $v$  at the time of the insertion. An insertion of an edge  $e = (v, u)$  into the set  $T(v)$  will cause  $v$  to change its label to the label of  $u$  plus  $n$ , that is  $P(u) + n$ . The edge  $e$  will also be labeled by this label.

In addition, for every vertex  $v$  a table  $M(v)$  is maintained. These tables are initially empty. Each table  $M(v)$  is used to store all the base values of levels  $P$  such that there exists at least one neighbor  $z$  of  $v$  that was labeled by  $P$  at some point of the execution of the algorithm, and such that the edge  $(v, z)$  was inserted into the set  $X(v)$  at that point of the execution.

The algorithm itself is very simple. It iteratively invokes the Procedure *Read\_Edge* on every edge of the stream, until the stream is exhausted. At this point it outputs the set  $\bigcup_v Sp(v) = \bigcup_v T(v) \cup \bigcup_v X(v)$  as the resulting spanner. The Procedure *Read\_Edge* accepts as input an edge  $e = (u, v)$  that it is supposed to “read”. The procedure finds the endpoint  $x$  of the edge  $e$  that has a greater label  $P(x)$  (with respect to the order relation  $\succ$ ). Suppose without loss of generality that  $x = u$ , i.e.,  $P(u) \succ P(v)$ . Then the procedure tests whether  $P(u)$  is a selected label. If it is, the edge  $e$  is inserted into the set of tree edges  $T(v)$  of  $v$ , and  $v$  adapts the label  $P(u) + n$ . If  $P(u)$  is not a selected label, then the procedure tests whether the base value  $B(P(u))$  of the label  $P(u)$  is stored in the table  $M(v)$ . If it is not, then the edge  $e$  is inserted into the set  $X(v)$  of the cross edges of  $v$ , and the label of  $v$  does not change. If  $P(u)$  is already stored in  $M(v)$  then nothing needs to be done.

The pseudo-code of the Procedure *Read\_Edge* is provided below. Its main difference from the description above is that the sets  $X(v)$  and  $T(v)$  are not maintained explicitly, but rather instead there is just one set  $Sp(v)$  maintained. The reason for this difference is that we aim to present the simplest version of the algorithm for which we can prove the desired bounds. However, it is more convenient to reason about the sets  $X(v)$  and  $T(v)$  explicitly, rather than about the set  $Sp(v)$  as a whole, and thus in the analysis we will analyze the version of the algorithm that maintains the sets  $T(v)$  and  $X(v)$  explicitly. (It is obvious that the two versions are equivalent.)

**Algorithm 1.** The streaming algorithm for constructing a sparse  $(2t - 1)$ -spanner, and Procedure *Read\_Edge*( $e = (u, v)$ ).

1. For all the edges  $e$  of the input stream  
    invoke *Read\_Edge*( $e$ )
2. Let  $u$  be the vertex s.t.  $P(u) \succ P(v)$
3. If ( $P(u)$  is a selected label) then
  - $P(v) \leftarrow P(u) + n$
  - $Sp(v) \leftarrow Sp(v) \cup \{e\}$
 else if ( $B(P(u)) \notin M(v)$ ) then
  - $M(v) \leftarrow M(v) \cup \{B(P(u))\}$
  - $Sp(v) \leftarrow Sp(v) \cup \{e\}$
 end-if

The set  $T(v)$  (resp.,  $X(v)$ ) is the set of edges inserted into the set  $Sp(v)$  on line 4 (resp., 7) of the Procedure *Read\_Edge*. We will say that an edge  $e$  is *inserted into  $T(v)$*  (resp.,  *$X(v)$* ) if it is inserted into  $Sp(v)$  on line 4 (resp., 7) of the algorithm.

Note that the Procedure *Read\_Edge* is extremely simple, and the only operations that might require a super-constant time are lines 5 and 6, which require testing a membership of an element in a data structure, and an insertion of an element into the data structure if it is not there already. These operations can be implemented very efficiently in a general scenario via a balanced search tree, or a hash table. Moreover, we will also show later that with high probability,

the size of each table is quite small, specifically  $\tilde{O}(n^{1/t})$ , and thus, in our setting these operations can be implemented even more efficiently.

### 2.2 The Size of the Spanner

We start with showing that the resulting spanner is sparse. For this end we show that both sets  $\bigcup_{v \in V} T(v)$  and  $\bigcup_{v \in V} X(v)$  are sparse.

**Lemma 2.** *For every vertex  $v \in V$ ,  $|T(v)| \leq t - 1$ .*

*Proof.* Each time an edge  $e = (v, u)$  is inserted into  $T(v)$ , the label of  $v$  grows from  $P(v)$  to  $P(u) + n$ . Moreover, note that  $P(u) \geq P(v)$  for such an edge. Consequently, the level of  $P(v)$  grows at least by 1. Hence at any given time of an execution of the algorithm,  $L(P(v))$  is an upper bound on the number of edges currently stored in  $T(v)$ . Since  $L(P(v))$  never grows beyond  $t - 1$ , it follows that  $|T(v)| \leq t - 1$ . ■

Consequently, the set  $\bigcup_v T(v)$  contains at most  $n \cdot (t - 1)$  edges, i.e.,

$$\left| \bigcup_{v \in V} T(v) \right| \leq n \cdot (t - 1). \tag{1}$$

We next argue that the set  $\bigcup_{v \in V} X(v)$  is sparse as well. First, by Lemma 1, the number of distinct labels of level  $t - 1$  that occur during the algorithm is, with high probability,  $O(n^{1/t} \cdot (t \cdot \log n)^{1-1/t})$ . Fix a vertex  $v \in V$ . Since, by line 5 of Algorithm 1 for each such a label  $P$  at most one edge  $(u, v)$  with  $P(u) = P \succ P(v)$  is inserted into  $X(v)$ , it follows that the number of edges  $(u, v)$  with  $P(u) \succ P(v)$ ,  $L(P(u)) = t - 1$ , inserted into  $X(v)$ , is, with high probability, at most  $O(n^{1/t} \cdot (t \cdot \log n)^{1-1/t})$ .

For an index  $i \in [(t - 1)]$ , let  $X^{(i)}(v)$  denote the set of edges  $(u, v)$ , with  $L(P(u)) < t - 1$ , inserted into  $X(v)$  during the period of time that  $L(P(v))$  was equal to  $i$ .

**Lemma 3.**  $X^{(t-1)}(v) = \emptyset$ .

Cardinalities of the sets  $X^{(i)}(v)$ ,  $0 \leq i \leq t - 2$ , are small as well. (Though these sets may be not empty.)

**Lemma 4.** *For every input sequence of edges  $(e_1, e_2, \dots, e_m)$  determined obliviously of the coin tosses of the algorithm, for every vertex  $v$ , and index  $i \in [(t - 2)]$ , with high probability,  $|X^{(i)}(v)| = O\left(n^{1/t} \cdot \frac{\log^{1-1/t} n}{t^{1/t}}\right)$ .*

We are now ready to state the desired upper bound on  $|\bigcup_{v \in V} X(v)|$ .

**Corollary 1.** *Under the assumption of Lemma 4, for every vertex  $v \in V$ , with high probability, the overall number of edges inserted into  $X(v)$  is  $O(n^{1/t} \cdot (t \cdot \log n)^{1-1/t})$ .*

We summarize the size analysis of the spanner constructed by Algorithm 1 with the following corollary.

**Corollary 2.** *Under the assumptions of Lemma 4 with high probability, the spanner  $H$  constructed by the algorithm contains  $O(n^{1+1/t} \cdot (t \cdot \log n)^{1-1/t})$  edges. Moreover, each table  $M(v)$ ,  $v \in V$ , stores, with high probability, at most  $O(n^{1/t} \cdot (t \cdot \log n)^{1-1/t})$  values, and consequently, overall the algorithm uses  $O(|H| \cdot \log n) = O(n^{1+1/t} \cdot t^{1-1/t} \cdot (\log n)^{2-1/t})$  bits of space.*

*Proof.* The resulting spanner is  $(\bigcup_{v \in V} T(v) \cup \bigcup_{v \in V} X(v))$ . By the inequality (II),  $|\bigcup_{v \in V} T(v)| \leq n \cdot (t-1)$ . By Corollary I, with high probability,  $|\bigcup_{v \in V} X(v)| = O(n^{1+1/t} \cdot (t \cdot \log n)^{1-1/t})$ , and so the first assertion of the corollary follows.

For the second assertion recall that a new value is added to  $M(v)$  only when a new edge  $(u, v)$  is introduced into the set  $X(v)$ . By Corollary I, with high probability,  $|X(v)| = O(n^{1/t} \cdot (t \cdot \log n)^{1-1/t})$ , and therefore the same bound applies for  $|M(v)|$  as well.

To calculate the overall size of the data structures used by the algorithms we note that  $|\bigcup_{v \in V} M(v)| \leq |\bigcup_{v \in V} X(v)| \leq |\bigcup_{v \in V} X(v)| + |\bigcup_{v \in V} T(v)| = O(|H|)$ . Since each label and edge requires  $O(\log n)$  bits to represent, the desired upper bound on the size of the data structures follows. ■

### 2.3 The Stretch Guarantee of the Spanner

We next show that the subgraph constructed by the algorithm is a  $(2t - 1)$ -spanner of the original graph  $G$ .

For an integer  $k \geq 1$ , and a vertex  $v \in V$ , let  $P_k(v)$  denote the label of  $v$ ,  $P(v)$ , before reading the  $k$ th edge of the input stream.

**Lemma 5.** *Let  $v, v' \in V$  be a pair of vertices such that there exist positive integers  $k, k' \geq 1$  such that  $B(P_k(v)) = B(P_{k'}(v'))$ . Then there exists a path of length at most  $L(P_k(v)) + L(P_{k'}(v'))$  between  $v$  and  $v'$  in the (final) set  $\bigcup_{v \in V} T(v)$ .*

The next lemma shows that the edge set  $H = \bigcup_{v \in V} T(v) \cup \bigcup_{v \in V} X(v)$  is a  $(2t - 1)$ -spanner.

**Lemma 6.** *Let  $e = (v, v') \in E$  be an edge. Then there exists a path of length at most  $2t - 1$  between  $v$  and  $v'$  in the edge set  $H$ .*

### 2.4 The Processing Time-per-Edge

To conclude the analysis of our streaming algorithm for constructing sparse spanners, we show that it has a very small processing time-per-edge. For this purpose we now fill in a few implementation details that have so far been unspecified. Specifically, on lines 5 and 6 of the Procedure *Read\_Edge* the algorithm tests whether an element  $B$  belongs to a set  $M(v)$ , and if it does not, the algorithm inserts it there. The set  $M(v)$  is a subset of the universe  $[n]$ , and by Corollary 2, its size is, with high probability,  $O((t \cdot \log n)^{1-1/t} \cdot n^{1/t})$ . Moreover, since  $|M(v)| \leq |X(v)|$ , it follows that  $|M(v)| \leq \deg(v)$ .

Let  $N = c \cdot (t \cdot \log n)^{1-1/t} \cdot n^{1/t}$ , for a sufficiently large constant  $c$ . (The probability that  $|M(v)| \leq c \cdot (t \cdot \log n)^{1-1/t} \cdot n^{1/t}$  for every vertex  $v \in V$  is at least  $1 - \frac{1}{n^{c-2}}$ . Hence choosing  $c = 4$  is sufficient.) As a part of preprocessing the algorithm computes a random hash function  $h : [n] \rightarrow [N]$ . Specifically, for each number  $i \in [n]$ , the algorithm picks a value  $j \in [N]$  uniformly at random, and sets  $h(i) = j$ . The table representation of this hash function is written down, and is used throughout the execution of the algorithm for the tables  $M(v)$  for all the vertices  $v \in V$ . This representation requires  $O(n \cdot \log n)$  space, and can be computed in  $O(n)$  time during the preprocessing.

For every vertex  $v$  the algorithm maintains a hash table  $M(v)$  of size  $N$ . Every base value  $B$  for which the algorithm needs to test its membership in  $M(v)$  on line 5 of the Procedure *Read\_Edge* is hashed to  $h(B)$  using the hash function  $h$ . To resolve collisions, for each entry of the hash table  $M(v)$  we use a dynamic dictionary data structure of Beame and Fich [13] (with the dynamization result of Andersson and Thorup [2]). This data structure maintains a dynamic set of  $q$  keys from an arbitrary universe using  $O\left(\sqrt{\frac{\log q}{\log \log q}}\right)$  time per update (insertion or deletion) and membership queries. This completes the description of the implementation details of the algorithm.

Note that the preprocessing of the algorithm requires  $O(n)$  time. In the full version of this paper we show that implemented this way, the algorithm enjoys an extremely low processing time-per-edge.

The properties of our streaming algorithm are summarized in the following theorem.

Let  $\lambda(n)$  (respectively,  $\sigma(n)$ ) denote the function  $\sqrt{\frac{\log \log n}{\log^{(3)} n}}$  (resp.,  $\sqrt{\frac{\log n}{\log \log n}}$ ).

**Theorem 1.** *Let  $n, t, n \geq t \geq 1$ , be positive integers. Consider an execution of our algorithm in the streaming model on an input (unweighted undirected)  $n$ -vertex graph  $G = (V, E)$  such that both the graph and the ordering  $\rho$  of its edges are chosen by a non-adaptive adversary obliviously of the coin tosses of the algorithm. The algorithm constructs a  $(2t - 1)$ -spanner  $H$  of the input graph. The expected size of the spanner is  $O(t \cdot n^{1+1/t})$  or the size of the spanner is  $O((t \cdot \log n)^{1-1/t} \cdot n^{1+1/t})$  with high probability (depending on the choice of  $p$ ; in the first case the guarantee on the size that holds with high probability is  $O(t \cdot \log n \cdot n^{1+1/t})$ ). The algorithm does so in one pass over the input stream, and requires  $O(1)$  expected processing time-per-edge,  $O(\lambda(n))$  processing time-per-edge with high probability, and  $O(\sigma(\deg(e)))$  processing time-per-edge in the worst-case, for an edge  $e = (v, u)$ . The space used by the algorithm is  $O(|H| \cdot \log n) = O(t^{1-1/t} \cdot \log^{2-1/t} n \cdot n^{1/t})$  bits with high probability. The preprocessing of the algorithm requires  $O(n)$  time.*

Our algorithm can be easily adapted to construct a  $(2t - 1)(1 + \epsilon)$ -spanners for weighted graphs, for an arbitrary  $\epsilon > 0$ . The size of the obtained spanner becomes  $O(\log_{(1+\epsilon)} \hat{\omega} \cdot (t \cdot \log n)^{1-1/t} \cdot n^{1+1/t})$ , where  $\hat{\omega}$  is the aspect ratio of the network. The algorithm still works in one pass, and has the same processing time-per-edge. This adaptation is achieved in a standard way (see, e.g., [20]), by

constructing  $\log_{(1+\epsilon)} \hat{\omega}$  different spanners in parallel. For completeness, we next overview this adaptation.

The edge weights can be scaled so that they are all greater or equal to 1 and smaller or equal to  $\hat{\omega}$ . All edges are partitioned logically into  $\lceil \log_{(1+\epsilon)} \hat{\omega} \rceil$  categories, indexed  $i = 1, 2, \dots, \lceil \log_{(1+\epsilon)} \hat{\omega} \rceil$ , according to their weights, with the category  $i$  containing the edges with weights greater or equal to  $(1 + \epsilon)^{i-1}$  and smaller than  $(1 + \epsilon)^i$ . When an edge  $e = (u, v)$  is read, it is processed according to its category, and it is either inserted into the spanner for the edges of category  $i$ , or discarded.

Obviously, after reading all the edges, we end up with  $\lceil \log_{(1+\epsilon)} \hat{\omega} \rceil$  subgraphs, with the  $i$ th subgraph being a  $(2t - 1)(1 + \epsilon)$ -spanner for the edges of the category  $i$ . Consequently, the union of all these edges is a  $(2t - 1)(1 + \epsilon)$ -spanner for the entire graph. The cardinality of this union is at most  $\lceil \log_{(1+\epsilon)} \hat{\omega} \rceil$  times the maximum cardinality of one of these subgraphs, which is, in turn, at most  $O((t \cdot \log n)^{1-1/t} \cdot n^{1+1/t})$  with high probability.

### 3 A Centralized Dynamic Algorithm

Our streaming algorithm can be seen as an incremental dynamic algorithm for maintaining a  $(2t - 1)$ -spanner of size  $O((t \cdot \log n)^{1-1/t} \cdot n^{1+1/t})$  for unweighted graphs, where  $n$  is an upper bound on the number of vertices that are allowed to appear in the graph.

The initialization of the algorithm is as follows. Given a graph  $G = (V, E)$ , we run our streaming algorithm with the edge set  $E$ , where the order in which the edge set is read is arbitrary. As a result, the spanner, and the satellite data structures  $\{M(v), Sp(v) \mid v \in V\}$  are constructed. This requires  $O(|E|)$  expected time,  $O(|E| \cdot \lambda(n))$  time with high probability, and  $O(|E| \cdot \sigma(\Delta))$  in the worst-case, where  $\Delta$  is the maximum degree of a vertex in  $G$ .

Each edge that is added to the graph is processed using our streaming algorithm. The spanner and the satellite data structures are updated in expected  $O(1)$  time-per-edge,  $O(\lambda(n))$  time-per-edge with high probability, and  $O(\sigma(\Delta))$  time in the worst-case.

We next make the algorithm robust to decremental updates (henceforth, *crashes*) as well. Note that for an edge  $e = (v, u)$  to become a  $T$ -edge (an edge of  $\bigcup_{x \in V} T(x)$ ), it must hold that at the time that the algorithm reads the edge, the greater of the two labels  $P(u)$  and  $P(v)$  (with respect to the order relation  $\succ$ ) is selected. The probability of a label to be selected is at most  $p = \left(\frac{t \cdot \log n}{n}\right)^{1/t}$  if its level is smaller than  $t - 1$ , and is 0 otherwise. Hence the probability of  $e$  to become a  $T$ -edge is at most  $p$ .

In the companion paper [16] it is shown that a crash of an edge  $e = (v, u)$  that does not belong to  $\bigcup_{x \in V} T(x)$  can be processed in expected time  $O(1)$ . Moreover, with high probability the processing of such a crash requires  $\sigma(h)$  time, where  $h = \max\{\deg(e), \log n\}$ . Since the entire spanner can be recomputed in expected time  $O(|E|)$  by our algorithm, it follows that the expected decremental update time of

our algorithm is  $O\left(\frac{|E|}{n^{1/t}} \cdot (t \cdot \log n)^{1/t}\right)$ . The size of the data structure maintained by the incremental variant of the algorithm is  $O(t^{1-1/t} \cdot (\log n)^{2-1/t} \cdot n^{1+1/t})$  bits, and the fully dynamic algorithm maintains a data structure of size  $O(|E| \cdot \log n)$ .

We summarize this discussion with a following corollary.

**Corollary 3.** *For positive integer  $n, t$ ,  $n \geq t \geq 1$ , the algorithm is a fully dynamic algorithm for maintaining  $(2t - 1)$ -spanners with expected  $O(t \cdot n^{1+1/t})$  number of edges (or  $O((t \log n)^{1-1/t} \cdot n^{1+1/t})$  edges with high probability, depending on the choice of  $p$ ) for graphs with at most  $n$  vertices. If  $G = (V, E)$  is the initial graph, then the initialization of the algorithm requires  $O(|E|)$  expected time,  $O(|E| \cdot \lambda(n))$  time with high probability, and  $O(|E| \cdot \sigma(\Delta))$  in the worst-case. The expected incremental update time of the algorithm is  $O(1)$ , with high probability it is  $O(\lambda(n))$ , and in the worst-case it is  $O(\sigma(\deg(e)))$  (for an edge  $e$  that joins the graph). The expected decremental update time is  $O\left(\frac{|E|}{n^{1/t}} \cdot (t \cdot \log n)^{1/t}\right)$ , and with probability at least  $1 - \left(\frac{t \cdot \log n}{n}\right)^{1/t}$  the decremental update time is  $O(\sigma(h))$ , where  $h = \max\{\deg(e), \log n\}$ .*

To our knowledge, this is the first fully dynamic algorithm for maintaining sparse spanners for a wide range of values of the stretch parameter  $t$  with non-trivial guarantees on both the incremental and decremental update times.

## References

1. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences* 58, 137–147 (1999)
2. Andersson, A., Thorup, M.: Tight(er) worst-case bounds on dynamic searching and priority queues. In: *Proc. of the 32nd Annual ACM Symp. on Theory of Computing*, pp. 335–342. ACM Press, New York (2000)
3. Ausillo, G., Franciosa, P.G., Italiano, G.F.: Small stretch spanners on dynamic graphs. In: *Proc. of the 13th European Symp. on Algorithms (ESA)*, pp. 532–543 (2005)
4. Awerbuch, B.: Complexity of network synchronization. *J. ACM* 4, 804–823 (1985)
5. Awerbuch, B., Kutten, S., Peleg, D.: Online load balancing in a distributed network. In: *Proc. 24th ACM Symp. on Theory of Comput.* pp. 571–580. ACM Press, New York (1992)
6. Awerbuch, B., Peleg, D.: Network synchronization with polylogarithmic overhead. In: *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pp. 514–522. IEEE Computer Society Press, Los Alamitos (1990)
7. Awerbuch, B., Peleg, D.: Routing with polynomial communication-space tradeoff. *SIAM J. Discrete Mathematics* 5, 151–162 (1992)
8. Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Reductions in streaming algorithms, with an applications to counting triangles in graphs. In: *Proc. 13th ACM-SIAM Symp. on Discr. Algor.* pp. 623–632. ACM Press, New York (2002)
9. Baswana, S.: personal communication (2006)
10. Baswana, S.: Dynamic algorithms for graph spanners. In: Azar, Y., Erlebach, T. (eds.) *ESA 2006*. LNCS, vol. 4168, Springer, Heidelberg (2006)

11. Baswana, S., Kavitha, T., Mehlhorn, K., Pettie, S.: New constructions of  $(a,b)$ -spanners and additive spanners. In: SODA: ACM-SIAM Symposium on Discrete Algorithms, pp. 672–681. ACM Press, New York (2005)
12. Baswana, S., Sen, S.: A simple linear time algorithm for computing a  $(2k - 1)$ -spanner of  $O(n^{1+1/k})$  size in weighted graphs. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 384–396. Springer, Heidelberg (2003)
13. Beame, P., Fich, F.E.: Optimal bounds for the predecessor problem. In: Proc. of the 31st Annual ACM Symp. on Theory of Computing, pp. 295–304. ACM Press, New York (1999)
14. Dor, D., Halperin, S., Zwick, U.: All-pairs almost shortest paths. *SIAM J. Comput.* 29, 1740–1759 (2000)
15. Elkin, M.: Computing almost shortest paths. In: Proc. 20th ACM Symp. on Principles of Distributed Computing, pp. 53–62. ACM Press, New York (2001)
16. Elkin, M.: A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. Manuscript (2006)
17. Elkin, M., Peleg, D.: Spanner constructions for general graphs. In: Proc. of the 33th ACM Symp. on Theory of Computing, pp. 173–182. ACM Press, New York (2001)
18. Elkin, M., Zhang, J.: Efficient algorithms for constructing  $(1 + \epsilon, \beta)$ -spanners in the distributed and streaming models. *Distributed Computing* 18, 375–385 (2006)
19. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. In: Proc. of the 31st International Colloq. on Automata, Languages and Progr. pp. 531–543 (2004)
20. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: Graph distances in the streaming model: The value of space. In: Proc. of the ACM-SIAM Symp. on Discrete Algorithms, pp. 745–754. ACM Press, New York (2005)
21. Feigenbaum, J., Strauss, S.K.M., Viswanathan, M.: An approximate  $L^1$  difference algorithm for massive data streams. *Journal on Computing* 32, 131–151 (2002)
22. Peleg, D., Ullman, J.D.: An optimal synchronizer for the hypercube. *SIAM J. on Comput.* 18, 740–747 (1989)
23. Peleg, D., Upfal, E.: A tradeoff between size and efficiency for routing tables. *J. of the ACM* 36, 510–530 (1989)
24. Pettie, S.: Ultrasparse spanners with sublinear distortion. Manuscript (2005)
25. Thorup, M., Zwick, U.: Approximate distance oracles. In: Proc. of the 33rd ACM Symp. on Theory of Computing, pp. 183–192. ACM Press, New York (2001)
26. Thorup, M., Zwick, U.: Compact routing schemes. In: Proc. of the 13th Symp. on Parallelism in Algorithms and Architectures, pp. 1–10 (2001)
27. Thorup, M., Zwick, U.: Spanners and emulators with sublinear distance errors. In: Proc. of Symp. on Discr. Algorithms, pp. 802–809 (2006)
28. Woodruff, D.: Lower bounds for additive spanners, emulators, and more. Manuscript (2006)



# Checking and Spot-Checking the Correctness of Priority Queues

Matthew Chu<sup>1</sup>, Sampath Kannan<sup>1</sup>, and Andrew McGregor<sup>2</sup>

<sup>1</sup> Dept. of Computer and Information Science, University of Pennsylvania  
`{mdchu,kannan}@cis.upenn.edu`

<sup>2</sup> ITA Center, University of California, San Diego  
`andrewm@ucsd.edu`

**Abstract.** We revisit the problem of memory checking considered by Blum et al. [3]. In this model, a checker monitors the behavior of a data structure residing in unreliable memory given an arbitrary sequence of user defined operations. The checker is permitted a small amount of separate reliable memory and must fail a data structure if it is not behaving as specified and pass it otherwise. How much additional reliable memory is required by the checker? First, we present a checker for an implementation of a priority queue. The checker uses  $O(\sqrt{n} \log n)$  space where  $n$  is the number of operations performed. We then present a *spot-checker* using only  $O(\epsilon^{-1} \log \delta^{-1} \log n)$  space, that, with probability at least  $1 - \delta$ , will fail the priority queue if it is  $\epsilon$ -far (defined appropriately) from operating like a priority queue and pass the priority queue if it operates correctly. Finally, we then prove a range of lower bounds that complement our checkers.

## 1 Introduction

Program checking [4] is a paradigm for gaining confidence at run-time in the output produced by a program by running an auxiliary program called the *checker* to verify the correctness of the output on the current input. Checkers are allowed to be probabilistic and have a specifiably small probability of themselves making a mistake; however, this probability of error depends only on the checker's internal coin tosses and not on the presence or absence of any particular bug in the program being checked. Checkers may also query the program being checked on additional inputs and use the self-consistency of the outputs to determine correctness. Checkers designed for a particular computational problem can check any program that claims to solve the problem.

Since checkers are run on-line, they should be efficient and should not introduce a significant overhead in the program's running time and resource use. Sometimes it is possible and desirable to design highly efficient checkers that only verify that the program output is *close* to the true output in a quantifiable way. Such checkers, known as *spot-checkers* [6] do not even look at the entire input and output of the program. Spot-checking is related to the idea of *property testing* [10] which seeks to examine small portions of the input to determine if it is close to having a specified property.

The problem of checking the correctness of the behavior of data structures residing in cheap but potentially unreliable memory using a small amount of reliable memory was first considered by Blum et al. [3]. In contrast to program checking, we emphasize that we are less concerned with ensuring that the data structure has been correctly coded but rather wish to ensure that the use of unreliable memory has not caused incorrect behavior. However, in the spirit of the program checking framework, we will assume a fully general scenario in which an adversary controls both the sequence of queries and updates to the data structure and the exact state of the data structure at each point.

A checker equipped with a small amount of reliable (but not necessarily secret) memory observes the sequence of operations and the results produced by the data structure and must decide whether it behaved correctly. Two flavors of checkers are considered in [3] — an *off-line* checker that makes its determination after an entire sequence of accesses have been made to the data structure, and an *on-line* checker that must immediately report an error when it occurs. These models were further explored in the context of simple linked data structures such as linked lists, trees, and graphs in a paper by Amato and Loui [2]. However, to date, no efficient checkers have been designed for data structures such as priority queues and search trees that maintain global order properties amongst the keys they store. We note that there has been some work done on designing new data structures that are resilient to memory faults, e.g., Finocchi et al. [9], but stress that this is a different problem than checking correctness.

The memory checkers of Blum et al. [3] and Amato and Loui [2] fit in a model of computation called the streaming model [11,11,7]. In this model a computer with a small amount of memory observes an adversarially generated stream and determines (with high probability) whether the stream satisfies a specified property. The notion of spot-checking in such a model [8] is a little different from the standard notion — rather than requiring the spot-checker to sample only a few places in the input and output, we allow the spot-checker to observe the entire stream of inputs and outputs, but require that it operate with the space constraints imposed by the streaming model.

A common implementation of priority queues is as binary heaps. A heap is a data structure that is used to easily remove the minimum key (min-heaps) or the maximum (max-heaps). For the purposes of this paper, we will consider min-heaps, but all conclusions can easily be applied to max-heaps as well. Heaps are usually implemented as binary trees, explicitly or implicitly as an array. Each node has a key, a right child, and a left child. There are two main properties of all heaps. The first is that all descendants of a certain node have a key that is greater than the node's key. The second property is that the tree is filled in order from left to right, level by level. All add and delete operations run in  $O(\log n)$  time, where  $n$  is the number of nodes in the tree.

## 1.1 Our Results

In this paper we present results on the checking and spot-checking of priority queues. We will assume that the priority queue is implemented in unreliable

memory and that the checker observes the values inserted into the priority queue and the result of each extract operation. We will present our results by referring to a binary-heap used to implement the priority queue but our checkers will be able to check any implementation. Note that our goal is only to verify that the input/output behavior of the implementation is correct. We do not (and in this model, cannot) actually verify that the priority queue is implemented as a heap and that this heap satisfies the structure property.

Our checkers are all off-line and assume that the sequence of operations start and end with the empty heap. We present a  $O(\sqrt{n} \log n)$ -space checker and an  $O(\epsilon^{-1} \log \delta^{-1} \log n)$ -space spot-checker in Section 3 and Section 4 respectively. In Section 5 we present lower bounds that show that any on-line checker requires  $\Omega(n/\log n)$  space, that any deterministic checker requires  $\Omega(n)$  space, and that our checker is near-optimal among checkers of a certain type.

## 2 Preliminaries

We start by adapting the definition of a memory checker from [3] for the checking of heaps. See Fig. 1 for an accompanying figure.

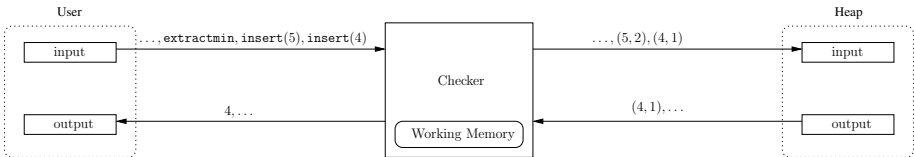


Fig. 1. Memory Checker interaction with User and Data Structure

**Definition 1 (Heap Checker).** A heap checker for heap  $\mathcal{H}$  is an (probabilistic) algorithm  $C$  with access to the following four tapes:

1. A “user” input tape from which the checker reads user specified operations. Each operation is either an `insert( $\cdot$ )` where the argument is a value to be inserted or an `extractmin`.
2. A “user” output tape on which the checker writes a value for each `extractmin` operation or alternatively writes `FAIL` and terminates if the checker has determined that the heap checker is not operating correctly. If the checker never outputs `FAIL` during the processing of the user specified operations, then the checker concludes by writing `PASS` on this tape.
3. A “heap” input tape on which the checker specifies operations to  $\mathcal{H}$ . On a user operation `insert( $u$ )`, the checker requests the insertion of the tuple  $(u, t)$  into the heap where  $t$  is the index of the user operation [1], referred to as the time-stamp. On a user `extractmin` operation, the checker requests that the heap extracts the smallest value presently stored in the heap.

<sup>1</sup> In the actual implementation of the heap the  $t$  is encoded as lower order bits and  $u$  is encoded as higher order bits. A consequence of this is that if  $(u, t)$  and  $(u, t')$  are concurrently in memory, then  $(u, t)$  should be extracted before  $(u, t')$  if  $t < t'$ .

4. A “heap” output tape from which the checker reads the output of each operation. The user `extractmin` operation will correspond to a tuple  $(u, t)$  being read from the heap.

In addition, the checker has a limited amount of working memory.

We can abstract the input of the heap checker as a sequence that encodes both the user specified operations and the output of the heap. We define this sequence and what it means for the sequence to be “heap-like” as follows.

**Definition 2 (Interaction Sequence and Heap-like).** An interaction sequence is a length  $2n$  sequence of tuples  $S = c_1 \dots c_{2n}$  where each tuple  $c_j$  is either an `insert` of some value  $u$ , denoted  $\langle \text{insert}, u \rangle$ , or an `extractmin` operation that returns some value  $v$  that purports to have inserted at time  $t$ , denoted  $\langle \text{extractmin}, (v, t) \rangle$ . Furthermore, we require that for any  $k \in [2n]$ , the number of `extractmin` operations among  $c_1 \dots c_k$  is at most the number of `insert` operations. We define an ordering of the tuples in the natural way:  $(u, t) < (u', t')$  iff  $u < u'$  or  $(u = u'$  and  $t < t')$ . We say an interaction sequence is heap-like if for all `extractmin` operations  $c_j$

$$c_j = \langle \text{extractmin}, (u, t) \rangle \Rightarrow (u, t) = \min M_{j-1} \quad , \quad (\text{C})$$

where,

$$M_0 = \emptyset \text{ and } M_j = \begin{cases} M_{j-1} \setminus \min M_{j-1} & \text{if } c_j = \langle \text{extractmin}, (\cdot, \cdot) \rangle \\ M_{j-1} \cup \{(u, j)\} & \text{if } c_j = \langle \text{insert}, u \rangle \end{cases} .$$

Conceptually,  $M_j$  is the set of (value, time-stamp) tuples still to be extracted from the heap if the heap were operating correctly.

It will be convenient for us to decompose the heap-like condition into three separate conditions in the case that we start and end with an empty heap. We prove these three condition are together equivalent in the following lemma.

**Lemma 1 (Equivalent Definition for Heap-like).** An interaction sequence  $S$  is heap-like iff  $S$  satisfies the following three conditions,

$$\{(u, t) : c_t = \langle \text{insert}, u \rangle\} = \{(u, t) : \langle \text{extractmin}, (u, t) \rangle \in S\} \quad , \quad (\text{C1})$$

$$\forall c_{t_b} = \langle \text{extractmin}, (\cdot, t_a) \rangle, t_a < t_b \quad , \quad \text{and} \quad (\text{C2})$$

$$\forall c_{t_b} = \langle \text{extractmin}, (u, t_a) \rangle, c_{t_{b'}} = \langle \text{extractmin}, (u', t_{a'}) \rangle, \\ (u, t_a) < (u', t_{a'}) \Rightarrow (t_{b'} < t_a \text{ or } t_b < t_{b'}) \quad . \quad (\text{C3})$$

These conditions correspond to the fact that 1) the set of (value, time-stamp) pairs inserted should match the (value, time-stamp) pairs extracted, 2) a (value, time-stamp) should only be extracted after it has been inserted and 3) a pair  $(u', t')$  should not be extracted between the insert and extraction of a  $(u, t)$  pair if  $(u, t) < (u', t')$ .

*Proof.* If  $S$  satisfies (C) then clearly (C3), (C2), and (C1) are satisfied. Conversely assume  $S$  satisfies (C3), (C2), and (C1). For the sake of contradiction, assume that there exists a  $t_{b'}$  such that,  $c_{t_{b'}} = \langle \text{extractmin}, (u', t_{a'}) \rangle$  and  $(u', t_{a'}) \neq \min M_{j-1}$ . and assume  $t_{b'}$  is the smallest such value. But then there exists  $(u, t_a) \in M_{j-1}$  with  $(u, t_a) < (u', t_{a'})$ . By the minimality of  $t_{b'}$  and (C1),  $t_a < t_{b'} < t_b$  for some  $t_b$  such that  $c_{t_b} = \langle \text{extractmin}, (u, t_a) \rangle$ . Hence  $S$  violates (C3) which is a contradiction. Hence  $S$  satisfies (C) after all.

Next we define what it means for a sequence to be far from heap-like. It will turn out that it is relatively straight-forward to ensure that a sequence satisfies (C2) and (C1). Hence, we only define distances between sequences that satisfy these two properties. Intuitively the distance between two interaction sequences  $S$  and  $S'$  will be the least number of moves that are necessary to transform  $S$  into  $S'$ . However, the exact definition is a little more awkward because of the (value, time-stamp) pairs in the `extractmin` operations.

**Definition 3 ( $\epsilon$ -far from Heap-like).** Consider an interaction sequence  $S = c_1c_2 \dots c_{2n}$  and let  $\sigma$  be a permutation on  $[2n]$ . Let  $S_\sigma = c'_1c'_2 \dots c'_{2n}$  be the sequence where,

$$c'_i = \begin{cases} \langle \text{insert}, u \rangle & \text{if } c_{\sigma^{-1}(i)} = \langle \text{insert}, u \rangle \\ \langle \text{extractmin}, (u, \sigma(t)) \rangle & \text{if } c_{\sigma^{-1}(i)} = \langle \text{extractmin}, (u, t) \rangle \end{cases} .$$

We define  $\text{dist}(S, S_\sigma)$  as the number of edits required to sort  $(\sigma(1), \dots, \sigma(2n))$  where an edit is of the form “move the value in position  $k$  and insert it at position  $j$ .” In particular we say  $S$  is  $\epsilon$ -far from being heap-like if for all  $\sigma$  such that  $S_\sigma$  is heap-like,  $\text{dist}(S, S_\sigma) \geq \epsilon n$ .

So, for example, for  $\sigma = (1 \rightarrow 1, 2 \rightarrow 3, 3 \rightarrow 4, 4 \rightarrow 2)$  and

$$S = \langle \text{insert}, 15 \rangle \langle \text{insert}, 16 \rangle \langle \text{extractmin}, (16, 2) \rangle \langle \text{extractmin}, (15, 1) \rangle$$

then,  $S_\sigma = \langle \text{insert}, 15 \rangle \langle \text{extractmin}, (15, 1) \rangle \langle \text{insert}, 16 \rangle \langle \text{extractmin}, (16, 3) \rangle$ . In this case,  $\text{dist}(S, S_\sigma) = 1$  and since  $S_\sigma$  is heap-like,  $S$  is at most  $1/2$ -far from being heap-like.

### 3 An $O(\sqrt{n} \log n)$ -Space Checker for Heaps

In this section we present a memory checker that accepts an interaction sequence that is heap-like and, with probability at least  $1 - \delta$ , rejects an interaction sequence that is not heap-like. The algorithm uses  $O(\sqrt{n} \log n + \log(1/\delta) \log n)$  space and processes each term of the interaction sequence in  $O(\log n + \log(1/\delta))$  time. Hence forth we assume that  $\delta > 1/n$  and therefore omit the  $\delta$  dependencies.

To ensure that the interaction sequence satisfies (C1), we use the  $\epsilon$ -biased hash function construction of Naor and Naor [12]. Their relevant properties are presented in the following theorem.

**Theorem 1 ( $\epsilon$ -biased Hash Function [12]).** *Consider two  $n$ -bit binary strings  $x$  and  $y$ . There exists a randomized scheme using  $O(\log n + \log \delta^{-1})$  random bits that constructs a hash function  $h$  such that  $\Pr(h(x) = h(y)) \leq \delta$  if  $x \neq y$ . Furthermore  $h(\cdot)$  can be computed in  $O(\log n + \log \delta^{-1})$  space even if the string to be hashed is revealed bit by bit in some arbitrary order.*

Using such a function we hash (value, time-stamp) pairs inserted and extracted to ensure that, with probability at least  $1 - \delta$ , condition [C1] is satisfied. It is easy to check that condition [C2] is also satisfied as for each operation  $c_{t_b} = (\text{extractmin}, (\cdot, t_a))$  it is sufficient to check that  $t_a < t_b$ .

To ensure condition [C3], the algorithm maintains two lists

$$E = \{(v_1, t_1), \dots, (v_{|E|}, t_{|E|})\},$$

a list of (value, time) pairs, and  $B$ , a list of recently inserted values and their time-stamps. The list  $E$  will define a series of epochs and  $B$  will be used to buffer (value, time) between the creation of new epochs. Specifically, the list  $E$  will be sorted such that  $t_1 < t_2 < \dots < t_{|E|}$ . We then refer to the period of time  $T_i = \{t : t_{i-1} < t \leq t_i\}$  as the  $i$ th epoch (where  $t_0 = 0$ ). We define an epoch to the period  $T_{|E|+1} = \{t : t > t_E\}$  as the *current epoch*.

The list  $B$  will contain all the values that have been inserted into the heap since the last pair was added to  $E$  not including those that have been returned by an `extractmin` operation. Together, the state of  $B$  and  $E$  at any point in the algorithm define the function,

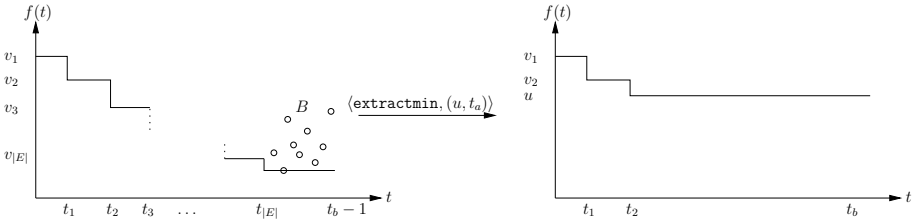
$$f(t) = \begin{cases} (v_i, t_i) & \text{if } t \in T_i \text{ for } i \leq |E| \\ \min B & \text{if } t \in T_{|E|+1} \end{cases} .$$

The semantics of  $f$  is such that at any time, if  $(u, t)$  has been inserted but not extracted, then,  $f(t) \leq (u, t)$  if the heap is performing correctly.  $f$  is potentially updated in every iteration of the algorithm. See Figure 2 for a schematic of the update and utility of the function  $f$ . The algorithm maintains  $B$  and  $E$  such that there are at most  $\sqrt{n}$  tuples in both sets. This is achieved by using the set  $B$  to, in effect, buffer inserted tuples such that at least  $\sqrt{n}$  tuples must be inserted for  $|E|$  to increase by 1. The algorithm is presented in Figure 3.

**Theorem 2.** *Algorithm [Heap-Checker] is a checker for the correctness of heaps. It uses  $O(\sqrt{n} \log n)$  memory and runs in  $O(\log n)$  time per term of the interaction sequence.*

*Proof.* We first prove the correctness of the [Heap-Checker] algorithm and then bound the space and time complexity.

**Correctness:** If  $S$  is heap-like then, the tester will PASS the sequence since property [C1] ensures that the algorithm does not fail in Line 11 and properties [C2] and [C3] ensure that the algorithm does not fail in Line 9. Conversely, if  $S$  does not satisfy property [C1] or [C2], the checker will fail the sequence at Line 11 or 9. Assuming that  $S$  has properties [C1] and [C2], we now show



**Fig. 2.** A schematic depicting the part of the behavior of the algorithm **Heap-Checker** when processing item  $c_{t_b} = \langle \text{extractmin}, (u, t_a) \rangle$  where  $t_2 < t_a < t_3$ . First it is checked that  $f(t_a) \leq (u, t_a)$  and, if so,  $\{(v_i, t_i) \in E : v_i < u\}$  is removed from  $E$  and  $(u, t_b)$  is added. Furthermore  $B$  is emptied.

that the checker will fail at Line **9** at some point during the processing of the interaction sequence. If property **(C3)** is not satisfied then consider the smallest  $t_b$  such that  $c_{t_b} = \langle \text{extractmin}, (u, t_a) \rangle$  and there exists a  $t_{b'}$  with  $c_{t_{b'}} = \langle \text{extractmin}, (u', t_{a'}) \rangle$  with  $(u, t_a) < (u', t_{a'})$  and  $t_a < t_{b'} < t_b$ . We consider two cases:

1. At time  $t_{b'}$  assume  $t_a$  is in the current epoch. Therefore  $(u, t) \in B$  and hence  $(u', t') > (u, t) \geq \min B$ . Therefore the algorithm fails in iteration  $t_{b'}$  at line **9**.
2. Otherwise assume that, at the start of iteration  $t_{b'}$ ,  $t_a$  is not in the current epoch, i.e.  $t_a < t_{|E|}$ . Therefore, at the end of the iteration  $v_{|E|} \geq u'$ . But then at the start of iteration  $t_b$ ,  $f(t_a)$  is also at least  $u'$ . Since at iteration  $t_b$  we have  $f(t_a) \geq u' > u$ , the algorithm fails at this iteration during Line **9**.

Space Use: It is clear that there are never more than  $\sqrt{n}$  values stored in  $B$ .

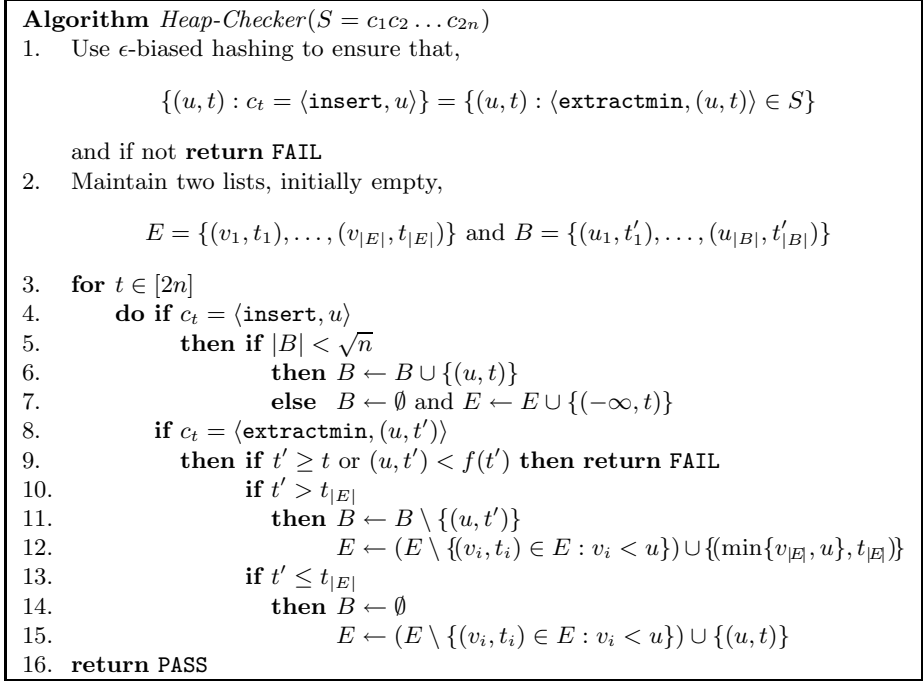
To bound the size of  $E$  we consider the following two ways in which a pair is added to  $E$ .

1. Line **7**: Note that there can be at most  $\sqrt{n}$  addition since in between each such addition there must be at least  $\sqrt{n}$  insert operations.
2. Line **15**: Note that there is no net increase in the size of  $E$  in this step. Therefore, the maximum number of pairs stored in  $E$  is  $\sqrt{n}$ . Hence the space use of the algorithm is  $O(\sqrt{n} \log n)$  as claimed.

Running Time: For keeping track of the current epoch, the checker can keep a heap of the values in  $B$  in its own reliable memory. This means that all insert and delete operations can be done in  $O(\log n)$  time. When updating the tuples in  $E$  the checker can do a binary search ( $O(\log n)$  time) through the values in each tuple since these are in sorted order. Since these are the only operations the checker must perform, it runs in  $O(\log n)$  time.

## 4 Spot-Checker

In this section we present a memory checker that accepts an interaction sequence that is heap-like and with probability at least  $1 - \delta$  rejects an interaction sequence



**Fig. 3.** The Heap-Checker Algorithm

that is  $\epsilon$ -far from being heap-like. The algorithm uses  $O(\epsilon^{-1} \log(1/\delta) \log n)$  space and processes each term of the interaction sequence in  $O(\epsilon^{-1} \log(1/\delta))$  time.

As before, we use the hashing techniques described in Section 3 to ascertain (with probability at least  $1 - \delta/2$ ) whether the sequence has property (C1) and can simply check for property (C2) by checking that the time-stamp of each extracted value does not exceed the current time. To check for property (C3), the algorithm stores a set of  $p = \ln(2/\delta)/\epsilon$  (value, time-stamp) pairs that are chosen at random from all the  $n$  such pairs. The hope is that one of the pairs stored will reveal that an interaction is un-heaplike if this is indeed the case. We make the following definition to clarify this notion.

**Definition 4 (Revealing Tuples).** *We call a tuple  $(u, t_a)$  a revealing tuple if there exists  $t_b, t_{a'}, t_{b'}, u'$  such that,  $c_{t_a} = \langle \text{insert}, u \rangle$ ,  $c_{t_b} = \langle \text{extractmin}, (u, t_a) \rangle$ ,  $c_{t_{a'}} = \langle \text{insert}, u' \rangle$ ,  $c_{t_{b'}} = \langle \text{extractmin}, (u', t_{a'}) \rangle$ ,  $t_a < t_{b'} < t_b$ , and  $(u, t_a) < (u', t_{a'})$ .*

In the time between the insertion and the extraction of a tuple  $(u, t_a)$ , the algorithm checks that no extraction returns a value  $u' > u$ . If this ever occurs then the sequence is not heap-like because the sequence violates property (C3). The crux of the proof of correctness will be that interaction sequence has many revealing tuples if it is far from being heap-like. The following lemma asserts that this is indeed the case. The algorithm is presented in Figure 4.



```

Algorithm Heap-Spot-Checker( $S = c_1 c_2 \dots c_{2n}$ )
1. Use  $\epsilon$ -biased hashing to ensure
    
$$\{(u, t) : c_t = \langle \text{insert}, u \rangle\} = \{(u, t) : \langle \text{extractmin}, (u, t') \rangle \in S\}$$

2. Let  $R$  be a set of  $p = \ln(2/\delta)/\epsilon$  values chosen randomly from the set  $[n]$ 
3. Maintain  $P$ , a list of at most  $p$  (value, insertion time) tuples
4.  $i \leftarrow 1$ 
5. for  $t \in [2n]$ 
6.     if  $c_t = \langle \text{insert}, u \rangle$  for some  $u$ 
7.         then if  $i \in R$  then  $P \leftarrow P \cup \{(u, t)\}$ 
8.              $i \leftarrow i + 1$ 
9.     if  $c_t = \langle \text{extractmin}, (u, t') \rangle$  for some  $u$ 
10.        then if  $t' > t$  then return FAIL
11.            if  $\exists (v, t_a) \in P$  with  $(v, t_a) < (u, t')$  then return FAIL
12.            if  $(u, t') \in P$  then  $P \leftarrow P \setminus \{(u, t')\}$ 
13. return PASS
    
```

Fig. 4. The *Heap-Spot-Checker* Algorithm

**Lemma 2.** Assume  $S$  satisfies,  $(C1)$  and  $(C2)$ . Then, if  $S$  is  $\epsilon$ -far from being heap-like, there are at least  $\epsilon n$  revealing tuples.

*Proof.* Assume  $S$  is  $\epsilon$ -far from being heap-like. Let  $r$  be the number of revealing tuples in  $S$ . Consider the interaction sequence  $S$  and let  $t_{b'}$  be the smallest value such that  $c_{t_{b'}}$  does not satisfy the heap-condition  $(C)$ ; i.e.,  $c_{t_{b'}} = \langle \text{extractmin}, (u', t_{a'}) \rangle$  but  $(u', t_{a'}) \geq (u, t_a)$  where  $(u, t_a) = \min M_{t_{b'}-1}$ . Let  $t_b$  be such that  $c_{t_b} = \langle \text{extractmin}, (u, t_a) \rangle$ . But then  $(u, t_a)$  is a revealing tuple. Consider rearranging the terms of  $S$  by bringing  $c_{t_b}$  up to position  $t_{b'}$  (and adjusting the position of the other terms and the time-stamps according). We claim this reduces the number of revealing tuples by at least one. To see this note that  $(u, t_a)$  is no longer a revealing tuple. Furthermore, note that no other tuple has become revealing.

We repeat this process until there are no operations that do not satisfy the heap-condition. Note that we can do this at most  $r$  times since we decrease the number of revealing tuples by one each time. Hence  $S$  is at most distance  $r$  from being heap-like and therefore  $r \geq \epsilon n$ .

**Theorem 3.** Algorithm *Heap-Spot-Checker* **PASSES** heap-like sequences and **FAILS** sequences that are  $\epsilon$ -far from being heap-like with probability at least  $1 - \delta$ . It uses  $O(\epsilon^{-1} \log(\delta^{-1}) \log n)$  memory and runs in  $O(\epsilon^{-1} \log(\delta^{-1}))$  time.

*Proof.* If a sequence is heap-like, the tester returns **PASS** because there will be no revealing tuples. From Lemma 2, we know that if a heap is  $\epsilon$ -far from being heaplike, there are at least  $\epsilon n$  revealing tuples. Furthermore, if any revealing insertion is sampled then the spot-checker will **FAIL** the sequence. The probability that a revealing insertion is stored by the algorithm is at least,

$$1 - (1 - \epsilon)^{\epsilon^{-1} \ln(2/\delta)} \geq 1 - \delta/2 .$$

The space requirement is obvious since the algorithm samples at most  $\epsilon^{-1} \ln(2/\delta)$  triples. The running time per-element is also  $O(\epsilon^{-1} \ln(1/\delta))$  since at each `extractmin` operation, at most  $\epsilon^{-1} \ln(1/\delta)$  tuples need to be checked for a potential violation.

## 5 Lower Bounds

The checker presented in Section 3 was off-line and required randomization. It would be preferable if the checker could identify any error as soon as the error occurred and, ideally, the checker would be deterministic. We start this section by showing that any checker that had either of these properties would need almost as much reliable working space as the space used to store the data structure.

**Theorem 4.** *Any on-line checker that is correct with probability at least 3/4 requires  $\Omega(n/\log n)$  working space. Any deterministic off-line checker requires  $\Omega(n)$  working space.*

*Proof.* The proofs will be by reductions from the one-round communication complexity of `PREFIX` :  $\{0, 1\}^n \times \{0, 1\}^n \times [n] \rightarrow \{0, 1\}$  and `EQUALITY` :  $\{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$  where

$$\text{PREFIX}(x, y, j) = \begin{cases} 1 & \text{if } \forall i \in [j], x_i = y_i \\ 0 & \text{otherwise} \end{cases}$$

and `EQUALITY`( $x, y$ ) = `PREFIX`( $x, y, n$ ).

Suppose there exists an on-line checker  $\mathcal{C}$  that correctly identifies the first error in the operation of the heap with probability at least 3/4. Consider an instance of `PREFIX` where Alice has a binary string  $x \in \{0, 1\}^n$  and Bob has a binary string  $y \in \{0, 1\}^n$  and an index  $j \in [n]$ . Then let Alice run the  $\mathcal{C}$  on the sequence  $\langle \text{insert}, x_1 \rangle \langle \text{insert}, x_2 + 2 \rangle \dots \langle \text{insert}, x_n + 2(n - 1) \rangle$  and then communicate the memory state of  $\mathcal{C}$  to Bob. Bob instantiates  $\mathcal{C}$  with this memory state and continues running  $\mathcal{C}$  on the sequence

$\langle \text{extractmin}, (y_1, 1) \rangle \langle \text{extractmin}, (y_2, 2) \rangle \dots \langle \text{extractmin}, (y_n + 2(n - 1), n) \rangle$  .

Then `PREFIX`( $x, y, j$ ) = 1 iff the  $\mathcal{C}$  does not fail until after the  $j$ th `extractmin` operation. But it was shown by Chakribatri et al. [5] that Alice needs to send  $\Omega(n/\log n)$  bits if Bob is to determine the value of `PREFIX`( $x, y, j$ ) with probability at least 3/4. Hence, the checker requires  $\Omega(n/\log n)$  bits.

The proof for the second part of the theorem is similar: Alice has a binary string  $x \in \{0, 1\}^n$  and Bob has a binary string  $y \in \{0, 1\}^n$  and wishes to learn `EQUALITY`( $x, y$ ). Alice and Bob create sequences and use a deterministic off-line checker as before. Then `EQUALITY`( $x, y$ ) = 1 iff the checker does not fail. But it is known (e.g. Kushilevitz and Nisan [13]) that Alice needs to send a message of length  $\Omega(n)$  if Bob is to determine the value of `EQUALITY`( $x, y$ ) with zero error. Hence the memory state of the checker must require  $\Omega(n)$  bits.

In the remainder of this section we argue that any checker that operates by storing (value, time-stamp) pairs and their extraction times must use  $\Omega(\sqrt{n})$  space if it is to succeed with at least constant probability. This is even the case if properties (C1) and (C2) are guaranteed. Let  $k = \sqrt{n}$ . Consider the following probabilistic construction of an interaction sequence  $S$ .

1. For  $i \in [k]$ , let  $R_i$  be the range  $[100(i-1)k, 100ik]$  and let  $S_i = \{u_1^i, \dots, u_k^i\}$  be  $k$  random elements in range  $R_i$ . We order the elements of each  $S_i$  such that,

$$u_1^1 \leq u_2^1 \leq \dots \leq u_k^1 < u_1^2 \leq \dots \leq u_k^2 \dots$$

2. Let  $j_1, \dots, j_k$  be random elements in the range  $[k]$ .
3. Consider the sequence  $S_{\text{good}}$  described as follows. First we insert  $S_k$  in a random order and then we extract  $j_k$  values from the heap. We call these deletions the *immediate deletes at stage  $k$* . We then insert  $S_{k-1}$  and extract  $j_{k-1}$  values. We continue in this way until we insert  $S_1$  and then we extract all the remaining values until the heap is empty. Let  $S_{\text{bad}}$  be the sequence constructed from  $S_{\text{good}}$  by choosing  $i \in [k]$  at random and swapping the last value extracted in the immediate deletes at stage  $i$  with the extraction of the  $(j_i + 1)$ th smallest element of  $S_i$ . Call these values  $u$  and  $v$  respectively. By construction  $u < v$ .
4. Let  $S = S_{\text{good}}$  with probability  $1/2$  and  $S = S_{\text{bad}}$  otherwise.

Note that by definition  $S_{\text{good}}$  is heap-like and, while  $S_{\text{bad}}$  satisfies (C1) and (C2), it violates (C3). Hence the only way for an algorithm (that only stores and compares (value, time-stamp) pairs along with their extraction times) to recognize if  $S = S_{\text{bad}}$  is to either a) have  $(u, \cdot)$  stored in memory when  $(v, \cdot)$  is extracted or b) have memory of the extraction time of  $(v, \cdot)$  when  $(u, \cdot)$  is extracted. Unfortunately since  $i$  and  $j_i$  are chosen at random, unless the algorithm can store  $O(k)$  pairs and deletion times then the probability of this is  $o(1)$ .

## 6 Conclusions and an Open Question

In this paper we presented a checker and an spot-checker for a priority queue. Both of are very practical and could be used as a guarantee of correct memory behavior when attempting to utilize cheap memory that may be unreliable.

We complemented the checkers with space lower bounds that showed that on-line checking and deterministic checking were infeasible. We also showed that off-line, randomized checkers of a specific class, that included the checker presented, required almost as much space as that required by the checker presented. However, it is conceivable that a better checker may exist that did not belong to this class. We conjecture that this is not the case. Also, a general proof would be very interesting because it appears that the such a proof is not possible with known techniques such as a reducing from communication complexity results. The reason for this is that consecutive subsequences of the interaction sequence can not be generated independently if the interaction sequence is to be heap-like.

## References

1. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences* 58(1), 137–147 (1999)
2. Amato, N.M., Loui, M.C.: Checking linked data structures. In: FTCS, pp. 164–173 (1994)
3. Blum, M., Evans, W.S., Gemmell, P., Kannan, S., Naor, M.: Checking the correctness of memories. *Algorithmica* 12(2/3), 225–244 (1994)
4. Blum, M., Kannan, S.: Designing programs that check their work. *J. ACM* 42(1), 269–291 (1995)
5. Chakrabarti, A., Cormode, G., McGregor, A.: A near-optimal algorithm for computing the entropy of a stream. In: ACM-SIAM Symposium on Discrete Algorithms (2007)
6. Ergün, F., Kannan, S., Kumar, R., Rubinfeld, R., Viswanathan, M.: Spot-checkers. *J. Comput. Syst. Sci.* 60(3), 717–751 (2000)
7. Feigenbaum, J., Kannan, S., Strauss, M., Viswanathan, M.: An approximate  $L^1$  difference algorithm for massive data streams. *SIAM Journal on Computing* 32(1), 131–151 (2002)
8. Feigenbaum, J., Kannan, S., Strauss, M., Viswanathan, M.: Testing and spot-checking of data streams. *Algorithmica* 34(1), 67–80 (2002)
9. Finocchi, I., Grandoni, F., Italiano, G.F.: Resilient Search Trees. In: ACM-SIAM Symposium on Discrete Algorithms, ACM Press, New York (2007)
10. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. *J. ACM* 45(4), 653–750 (1998)
11. Henzinger, M.R., Raghavan, P., Rajagopalan, S.: Computing on data streams. Technical Report 1998-001, DEC Systems Research Center (1998)
12. Naor, J., Naor, M.: Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.* 22(4), 838–856 (1993)
13. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)

# Undecidability of 2-Label BPP Equivalences and Behavioral Type Systems for the $\pi$ -Calculus

Naoki Kobayashi and Takashi Suto

Tohoku University  
{koba,tsuto}@kb.ecei.tohoku.ac.jp

**Abstract.** The trace equivalence of BPP was shown to be undecidable by Hirshfeld. We show that the trace equivalence remains undecidable even if the number of labels is restricted to two. The undecidability result holds also for the simulation of two-label BPP processes. These results imply undecidability of some behavioral type systems for the  $\pi$ -calculus.

## 1 Introduction

BPP [2] is a process calculus which has prefixes ( $lP$ ), sum, parallel composition, and recursion as process constructors. Hirshfeld [3] has shown that the trace equivalence of two BPP processes is undecidable, by encoding the halting problem of a Minsky machine into the trace inclusion relation between two BPP processes. Hüttel [4] extended the undecidability result to other preorders between processes.

In this paper, we show that the trace inclusion of BPP processes remains undecidable even if we restrict the number of action labels to two. In the rest of the paper, we call the restriction of BPP to two labels *2-label BPP*. Hirshfeld's encoding of a Minsky machine requires 6 action labels, hence his result does not immediately extend to the case of 2-label BPP processes.

One may think that the undecidability for 2-label BPP processes can be easily obtained by encoding an action label into a sequence of the two labels, so that  $P \leq_{tr} Q$  if and only if  $\llbracket P \rrbracket \leq_{tr} \llbracket Q \rrbracket$ , where  $P \leq_{tr} Q$  means that the trace set of  $P$  is a subset of the trace set of  $Q$ , and  $\llbracket P \rrbracket$  is the 2-label BPP process obtained from  $P$  by using the label encoding. Then, the undecidability of the trace inclusion for 2-label BPP (and hence also the undecidability of the trace equivalence) would follow from the undecidability for general BPP processes. We basically follow this approach, but there are two main difficulties. First, because of the existence of parallel composition, encoding of some action of a process may be simulated by interleaving execution of encodings of two or more actions of the other process. For example, consider two processes  $P_1 = l_2 | l_2$  and  $Q_1 = (l_2 | l_2) + l_3 l_1 l_1$  and choose the following label encoding:  $\llbracket l_1 \rrbracket = a$ ,  $\llbracket l_2 \rrbracket = ba$ ,  $\llbracket l_3 \rrbracket = bb$ . Then, the trace sets of  $P_1$  and  $Q_1$  are of course different, but the trace sets of  $\llbracket P_1 \rrbracket = ba | ba$  and  $\llbracket Q_1 \rrbracket = (ba | ba) + bbaa$  are equivalent. Second, a naive encoding may also invalidate the equivalence of processes. For example, consider  $P_2 = l_2 | l_2$  and  $Q_2 = l_2 l_2$ . These have the same trace sets (and they are even bisimilar). With

the above encoding, however,  $\llbracket P_2 \rrbracket$  has the trace  $baaa$  while  $\llbracket Q_2 \rrbracket$  does not. To overcome the first problem, we choose an encoding of labels such that a shuffle of two or more encoded labels (i.e., a partial trace of  $\llbracket l_1 \rrbracket \mid \cdots \mid \llbracket l_m \rrbracket$ ) cannot be confused with encoding  $\llbracket l \rrbracket$  of a single action. To avoid the second problem, we prepare a process  $Inv$  that simulates invalid sequences. With  $Inv$ , we can establish that  $P \leq_{tr} Q$  if and only if  $\llbracket P \rrbracket \leq_{tr} \llbracket Q \rrbracket \mid Inv$ , since  $Inv$  simulates all the invalid sequences of  $\llbracket P \rrbracket$  (which are generated by interleaving execution of more than one encoded actions). A similar (but a little more complicated) technique can also be used to show the undecidability of the simulation preorder of 2-label BPP processes<sup>1</sup>

As an application of the undecidability results above, we show that the type checking problems for some behavioral type systems for the  $\pi$ -calculus are also undecidable<sup>2</sup>. In the behavioral type systems, channel types are augmented with *usage expressions* (usages, in short), describing how each communication channel is used. The usages can be regarded as 2-label BPP processes. Since the trace preorder between two usages can be reduced to the typability of a certain process, the type checking problem is undecidable.

The rest of this paper is structured as follows. Section 2 introduces BPP. Section 3 proves the undecidability of trace inclusion of 2-label BPP. Section 4 applies a similar technique to prove that the simulation preorder is also undecidable for 2-label BPP. Section 5 applies the undecidability results to show that certain behavioral type systems for the  $\pi$ -calculus are undecidable. Section 6 discusses related work, and Section 7 concludes. Full proofs are found in the full version of this paper [11].

## 2 Basic Parallel Processes (BPP)

BPP [2] is a calculus of processes consisting of prefixes, parallel composition, internal choice, and recursion. Unlike in CCS [13], there is no synchronization mechanism (such as the transition  $a.P \mid \bar{a}.Q \xrightarrow{\tau} P \mid Q$ ).

The syntax of processes is given by:

$$P ::= \mathbf{0} \mid X \mid lP \mid (P|Q) \mid P + Q \mid \mu X.P$$

Here,  $X$  and  $l$  are meta-variables ranging over the sets of *process variables* and *action labels* respectively. We write  $\mathbf{Act}$  for the set of action labels, and write  $\mathbf{BPP}_{\mathbf{Act}}$  for the set of BPP processes whose action labels are in the set  $\mathbf{Act}$ .

The process  $\mathbf{0}$  does nothing. A process  $lP$  first performs  $l$  and then behaves like  $P$ .  $P|Q$  is a parallel composition of  $P$  and  $Q$ , and  $P + Q$  is an internal choice of  $P$  or  $Q$ .  $\mu X.P$  stands for the recursive process  $X$  usually defined by the equation  $X = P$ .

<sup>1</sup> Note that the trace inclusion (as well as the simulation preorder) is decidable for 1-label BPP processes.

<sup>2</sup> Actually, investigation into the type checking problems lead us to the study of the trace and simulation preorders for 2-label BPP in this paper.

|   |           |  |           |
|---|-----------|--|-----------|
| $\frac{}{lP \xrightarrow{l} P}$                         | (TR-ACT)  | $\frac{[\mu X.P/X]P \xrightarrow{l} Q}{\mu X.P \xrightarrow{l} Q}$ | (TR-REC)  |
| $\frac{P \xrightarrow{l} P'}{P Q \xrightarrow{l} P' Q}$ | (TR-PARL) | $\frac{Q \xrightarrow{l} Q'}{P Q \xrightarrow{l} P Q'}$            | (TR-PARR) |
| $\frac{P \xrightarrow{l} P'}{P+Q \xrightarrow{l} P'}$   | (TR-ORL)  | $\frac{Q \xrightarrow{l} Q'}{P+Q \xrightarrow{l} Q'}$              | (TR-ORR)  |

**Fig. 1.** Transition rules of BPP processes

We often omit  $\mathbf{0}$  and just write  $a$  for  $a\mathbf{0}$ . We give a higher-precedence to unary prefixes,  $+$ , and  $|$  in this order, so that  $l_1P_1|l_2P_2 + l_3P_3$  means  $(l_1P_1)|((l_2P_2) + (l_3P_3))$ .

We say that  $P$  is guarded by  $l$  in  $lP$ . A recursive process  $\mu X.P$  is *guarded* if  $X$  appears only in guarded positions of  $P$ . A process is *guarded* if all its recursive processes are guarded. In the rest of this paper, we consider only closed, guarded processes<sup>3</sup>

The transition relation  $P \xrightarrow{l} Q$  is the least relation closed under the rules in Figure 1. We write  $P \xrightarrow{l_1 \dots l_n} Q$  if  $P \xrightarrow{l_1} \dots \xrightarrow{l_n} Q$ .

*2-label BPP* is BPP where the set **Act** of action labels is restricted to the set  $\{a, b\}$ . Hence, the set of 2-label BPP processes is **BPP** <sub>$\{a,b\}$</sub> .

### 3 Undecidability of Trace Equivalence

In this section, we show that the trace equivalence of 2-label BPP processes is undecidable. As sketched in Section 2, we show an encoding of general BPP processes into 2-label BPP processes, so that the trace preorder is preserved. Then, the undecidability follows from the undecidability result for general BPP [3]. The undecidability of the trace equivalence can be shown also by using the encoding in Section 4, but the encoding presented in this section is simpler and easier to understand.

#### 3.1 Trace Set, Trace Preorder, and Trace Equivalence

**Definition 1 (trace set).** The *trace set* of  $P$ , written  $traces(P)$ , is defined by:

$$traces(P) = \{l_1 \dots l_n \mid P \xrightarrow{l_1} \dots \xrightarrow{l_n} P_n\}$$

**Definition 2.** The *trace preorder*  $\leq_{tr}$  and the *trace equivalence*  $\sim_{tr}$  are defined by:

$$P \leq_{tr} Q \stackrel{def}{\iff} traces(P) \subseteq traces(Q)$$

<sup>3</sup> Actually, any recursive process can be transformed to a bisimilar, guarded recursive process. For example,  $\mu X.(X|X)$  is equivalent to the guarded process  $\mu X.l(X|X)$ .  $\mu X.X$  is bisimilar to  $\mathbf{0}$ .

$$P \sim_{tr} Q \stackrel{def}{\iff} P \leq_{tr} Q \wedge Q \leq_{tr} P$$

### 3.2 Encoding

We first define the encoding of labels. Since the number of labels occurring in a given process is finite, we assume here that the set **Act** of action labels is a finite set  $\{l_0, \dots, l_{N-1}\}$ . In the rest of this section and Section 4, we use meta-variables  $P, Q, \dots$  for processes in  $\mathbf{BPP}_{\{l_0, \dots, l_{N-1}\}}$ , and use meta-variables  $E, F, \dots$  for processes in  $\mathbf{BPP}_{\{a, b\}}$ .

**Definition 3.** A mapping  $\llbracket \cdot \rrbracket$  from **Act** to  $\{a, b\}^*$  is defined by:

$$\llbracket l_i \rrbracket = ab^i ab^{2N-1-i}$$

Here,  $a^i$  stands for the sequence of  $a$  of length  $i$ . For example,  $a^3 = aaa$ .

We now define encoding of a process. As mentioned in Section 1, we use different encodings for  $P$  and  $Q$  in  $P \leq_{tr} Q$ .

#### Definition 4

Mappings  $\llbracket \cdot \rrbracket_L$  and  $\llbracket \cdot \rrbracket_R$  from  $\mathbf{BPP}_{\{l_0, \dots, l_{N-1}\}}$  to  $\mathbf{BPP}_{\{a, b\}}$  are defined by:

$$\begin{aligned} \llbracket \mathbf{0} \rrbracket_L &= \mathbf{0} & \llbracket P \mid Q \rrbracket_L &= \llbracket P \rrbracket_L \mid \llbracket Q \rrbracket_L \\ \llbracket X \rrbracket_L &= X & \llbracket P + Q \rrbracket_L &= \llbracket P \rrbracket_L + \llbracket Q \rrbracket_L \\ \llbracket lP \rrbracket_L &= \llbracket l \rrbracket \llbracket P \rrbracket_L & \llbracket \mu X.P \rrbracket_L &= \mu X. \llbracket P \rrbracket_L \\ \llbracket P \rrbracket_R &= \llbracket P \rrbracket_L \mid \text{Inv} \\ \text{where } \text{Inv} &= \sum_{k < N, k+l < 2N-1} ab^k ab^l aG \text{ and } G = \mu X.(aX + bX) \end{aligned}$$

The role of the process  $\text{Inv}$  in  $\llbracket P \rrbracket_R$  is to simulate invalid transition sequences (caused by interleaving execution of  $\llbracket l_i \rrbracket$  and  $\llbracket l_j \rrbracket$ ).

### 3.3 Undecidability of Trace Equivalence

The main result of this section is stated as follows.

**Theorem 1.**  $P \leq_{tr} Q$  if and only if  $\llbracket P \rrbracket_L \leq_{tr} \llbracket Q \rrbracket_R$ .

Since  $P \leq_{tr} Q$  is undecidable for general BPP [3], we obtain the following corollary.

**Corollary 1.** The trace inclusion  $E \leq_{tr} F$  and trace equivalence  $E \sim_{tr} F$  are undecidable for 2-label BPP.

*Proof.* If the trace inclusion  $\leq_{tr}$  were decidable for 2-label BPP, then we could decide  $P \leq_{tr} Q$  for general BPP by deciding  $\llbracket P \rrbracket_L \leq_{tr} \llbracket Q \rrbracket_R$ , hence a contradiction. To see that  $E \sim_{tr} F$  is also undecidable, it suffices to observe that  $E \leq_{tr} F$  if and only if  $E + F \sim_{tr} F$ .  $\square$

The rest of this section is devoted to the proof of Theorem 1. The followings are key lemmas needed to prove Theorem 1.



**Lemma 1.** *Let  $m \in \{L, R\}$ . If  $P \xrightarrow{l} Q$ , then  $\llbracket P \rrbracket_m \xrightarrow{\llbracket l \rrbracket} \llbracket Q \rrbracket_m$ .*

**Lemma 2.** *Let  $m \in \{L, R\}$ . If  $\llbracket P \rrbracket_m \xrightarrow{\llbracket l \rrbracket} E$ , then there exists a process  $Q$  such that  $E = \llbracket Q \rrbracket_m$  and  $P \xrightarrow{l} Q$ .*

Lemma 1, which follows by straightforward induction on the derivation of  $P \xrightarrow{l} Q$ , says that any transition of  $P$  can be simulated by  $\llbracket P \rrbracket_L$  and  $\llbracket P \rrbracket_R$ . Lemma 2 says that any *valid* (in the sense that the transition label sequence corresponds to a label of  $P$ ) transition sequence of  $\llbracket P \rrbracket_L$  or  $\llbracket P \rrbracket_R$  can be simulated by  $P$ .

Lemma 2 follows by induction on the derivation of the first transition of  $\llbracket P \rrbracket_m \xrightarrow{\llbracket l \rrbracket} E$ ; See 111 for the full proof of Lemma 2. The proof makes use of the following key property, which essentially says that the first problem mentioned in Section 1 (that a single action may be simulated by interleaving execution of two or more actions) cannot occur.

**Lemma 3.** *If  $\llbracket P_1 \mid P_2 \rrbracket_L \xrightarrow{\llbracket l \rrbracket} E$ , then either (i)  $\llbracket P_1 \rrbracket_L \xrightarrow{\llbracket l \rrbracket} E_1$  and  $E = E_1 \mid \llbracket P_2 \rrbracket_L$  or (ii)  $\llbracket P_2 \rrbracket_L \xrightarrow{\llbracket l \rrbracket} E_2$  and  $E = \llbracket P_1 \rrbracket_L \mid E_2$*

*Proof sketch.* By the transition rules, we have: (i)  $\llbracket P_1 \rrbracket_L \xrightarrow{s_1} E_1$ , (ii)  $\llbracket P_2 \rrbracket_L \xrightarrow{s_2} E_2$ , (iii)  $E = E_1 \mid E_2$ , and (iv)  $\llbracket l \rrbracket$  is a shuffle of  $s_1$  and  $s_2$ . It suffices to show that either  $s_1$  or  $s_2$  is an empty sequence. Suppose that  $s_1$  and  $s_2$  are not empty. Then  $s_1$  and  $s_2$  must be of the form  $ab^{j_1}$  and  $ab^{j_2}$  where  $j_1, j_2 \leq N - 1$ . Then,  $\llbracket l \rrbracket_L$  cannot be a shuffle of  $s_1$  and  $s_2$ , since  $\llbracket l \rrbracket_L$  contains  $2N - 1$  occurrences of  $b$ , whereas  $j_1 + j_2 \leq 2N - 2$ . □

We state another key lemma below. Let  $\mathbf{InvTr} = \{s \in \{a, b\}^* \mid \neg \exists s', l. (s = \llbracket l \rrbracket s')\}$ . In other words,  $\mathbf{InvTr}$  is the set of label sequences whose prefixes do not match  $\llbracket l \rrbracket$ .

**Lemma 4.** *If  $s \in \mathbf{InvTr} \cap \text{traces}(\llbracket P \rrbracket_L)$ , then  $s \in \text{traces}(Inv)$ .*

Lemma 4 means that any initially invalid sequence generated by  $\llbracket P \rrbracket_L$  can be simulated by  $Inv$ . Thus, the second problem mentioned in Section 1 is resolved.

We can now prove Theorem 1.

*Proof of Theorem 1*

- “Only if”: Suppose  $P \leq_{tr} Q$  and  $s \in \text{traces}(\llbracket P \rrbracket_L)$ . We need to show  $s \in \text{traces}(\llbracket Q \rrbracket_R)$ .  $s$  must be of the form  $\llbracket l_{k_1} \rrbracket \cdots \llbracket l_{k_n} \rrbracket s'$  where  $s' \in \mathbf{InvTr}$ . By Lemma 2, there exists  $P_1$  such that  $P \xrightarrow{l_{k_1} \cdots l_{k_n}} P_1$  and  $s' \in \text{traces}(\llbracket P_1 \rrbracket_L)$ . By the assumption, there must exist  $Q_1$  such that  $Q \xrightarrow{l_{k_1} \cdots l_{k_n}} Q_1$ . By using Lemma 1, we get  $\llbracket Q \rrbracket_R \xrightarrow{\llbracket l_{k_1} \cdots l_{k_n} \rrbracket} \llbracket Q_1 \rrbracket_R$ . By Lemma 4, we have  $s' \in \text{traces}(Inv) \subseteq \text{traces}(\llbracket Q_1 \rrbracket_R)$ . Thus, we have  $s \in \text{traces}(\llbracket Q \rrbracket_R)$  as required.
- “If”: Suppose  $\llbracket P \rrbracket_L \leq_{tr} \llbracket Q \rrbracket_R$  and  $l_{k_1} \cdots l_{k_n} \in \text{traces}(P)$ . By Lemma 1,  $\llbracket l_{k_1} \rrbracket \cdots \llbracket l_{k_n} \rrbracket \in \text{traces}(\llbracket P \rrbracket_L)$ . By the assumption  $\llbracket P \rrbracket_L \leq_{tr} \llbracket Q \rrbracket_R$ , we have  $\llbracket l_{k_1} \rrbracket \cdots \llbracket l_{k_n} \rrbracket \in \text{traces}(\llbracket Q \rrbracket_R)$ . By using Lemma 2, we obtain  $l_{k_1} \cdots l_{k_n} \in \text{traces}(Q)$  as required. □

### 4 Undecidability of Simulation Equivalence

In this section, we show that the simulation preorder and equivalence are also undecidable for 2-label BPP. We use the undecidability of the simulation preorder for the general BPP [4]<sup>4</sup>

**Definition 5.** A binary relation  $\mathcal{R}$  on BPP processes is a simulation if, for any  $P, Q, l$  such that  $P\mathcal{R}Q$  and  $P \xrightarrow{l} P'$ , there exists  $Q'$  such that  $Q \xrightarrow{l} Q'$  and  $P'\mathcal{R}Q'$ . The simulation preorder  $\leq_{sim}$  is the union of all simulations, i.e.,  $P \leq_{sim} Q$  if and only if there exists a simulation  $\mathcal{R}$  such  $P\mathcal{R}Q$ . We write  $P \sim_{sim} Q$  if  $P \leq_{sim} Q \wedge Q \leq_{sim} P$ .

Note that  $\leq_{sim}$  itself is a simulation (hence, the largest simulation).

We show the undecidability of the simulation preorder for 2-label BPP, by reduction of the simulation preorder for general BPP into that for 2-label BPP. We first need to change the encoding  $[\cdot]_R$  of the right-hand side process.

**Definition 6.** A mapping  $[\cdot]_{R'}$  from  $\mathbf{BPP}_{\{l_0, \dots, l_{N-1}\}}$  to  $\mathbf{BPP}_{\{a, b\}}$  is defined by:

$$\begin{aligned}
 [0]_{R'} &= 0 & [P]^{\epsilon, k_1, k_2} &= [P]_{R'} \\
 [X]_{R'} &= X & [P]^{as, 1, k} &= a[P]^{s, 2, k} + aH^{(2N-2-k)} \\
 [lP]_{R'} &= a[P]^{s, 1, 0} \quad (as = [l]) & [P]^{bs, 1, k} &= b[P]^{s, 1, k+1} + aH^{(2N-2-k)} \\
 [P | Q]_{R'} &= [P]_{R'} | [Q]_{R'} & [P]^{bs, 2, k} &= b[P]^{s, 2, k+1} + aG \\
 [P + Q]_{R'} &= [P]_{R'} + [Q]_{R'} & H^{(k)} &= \begin{cases} aG & (k = 0) \\ bH^{(k-1)} + aG & (k > 0) \end{cases} \\
 [\mu X.P]_{R'} &= \mu X.[P]_{R'}
 \end{aligned}$$

Note that the process  $G$  has been defined in Definition [4].

Intuitively,  $[P]^{s, k_1, k_2}$  represents an intermediate state for simulating a single action of the original process. The sequence  $s \in \{a, b\}^*$  is the remaining sequence of actions to be performed, and  $k_1$  and  $k_2$  are the numbers of  $a$  and  $b$  actions that have been already performed. The roles of  $aH^{(2N-2-k)}$  and  $aG$  in the definitions of  $[P]^{s, k_1, k_2}$  are to simulate invalid transitions.

**Theorem 2.**  $P \leq_{sim} Q$  if and only if  $[P]_L \leq_{sim} [Q]_{R'}$ .

To show the “if” part, it suffices to show that the relation  $\{(P, Q) \mid [P]_L \leq_{sim} [Q]_{R'}\}$  is a simulation. To show the “only if” part, we use the following, standard up-to technique:

**Lemma 5 (up-to technique).** Let  $\mathcal{R}$  be a binary relation on BPP processes. If  $\mathcal{R}$  satisfies:

$$\forall P, Q, P', l. ((P\mathcal{R}Q \wedge P \xrightarrow{l} P') \Rightarrow \exists Q'. (Q \xrightarrow{l} Q' \wedge P' \leq_{sim} \mathcal{R} \leq_{sim} Q')),$$

then  $\mathcal{R} \subseteq \leq_{sim}$ .

<sup>4</sup> The proofs in [4] contain some flaws, but the undecidability results are valid. Please refer to [12] for the flaws and corrected proofs of the undecidability for the general BPP.

To show the “only if” part of Theorem 2, it suffices to show that the following relation is a simulation up to  $\leq_{sim}$  (i.e., satisfies the assumption of Lemma 5).

$$\begin{aligned} \llbracket \leq_{sim} \rrbracket = & \{ (E, E) \mid E \in \mathbf{BPP}_{\{a,b\}} \} \\ & \cup \{ (\llbracket P \rrbracket_L, \llbracket Q \rrbracket_{R'}) \mid P \leq_{sim} Q \} \\ & \cup \{ (E, F) \mid P \leq_{sim} Q \wedge P' \leq_{sim} Q' \wedge s_1 s_2 = \llbracket U \rrbracket \wedge s_1, s_2 \neq \epsilon \wedge \\ & \quad \llbracket P \rrbracket_L \xrightarrow{s_1} E \xrightarrow{s_2} \llbracket P' \rrbracket_L \wedge \llbracket Q \rrbracket_{R'} \xrightarrow{s_1} F \xrightarrow{s_2} \llbracket Q' \rrbracket_{R'} \} \end{aligned}$$

$E$  and  $F$  in the third set are intermediate states for simulating a single action of general BPP processes. If  $E$  performs a valid action and becomes  $E'$ , then  $F$  can also perform a valid action to become  $F'$  so that the pair  $(E', F')$  is again in the second or third set. If  $E$  performs an invalid action to become  $E'$ , then  $F$  can transit to a process containing  $H^{(2N-2-k)}$  or  $G$ , which can simulate any transitions of  $E'$ . See [11] for the full proof.

As a corollary of the above theorem and the undecidability for general BPP [4,12], we obtain the undecidability for 2-label BPP.

**Corollary 2.** *The relations  $\leq_{sim}$  and  $\sim_{sim}$  are undecidable for 2-label BPP.*

## 5 Application to Behavioral Type Systems

In this section, we apply the undecidability results of the previous sections to show the undecidability of certain behavioral type systems for the  $\pi$ -calculus.

Behavioral type systems [6,10,11,9,16] use types to control how processes may interact with each other. They have been used for analyzing deadlocks, race conditions, and termination, etc. The version of behavioral type systems we discuss below is a type system with *channel usages* [14,10,9,8], which express how each communication channel is used for input and output.

### 5.1 Syntax of Usages, Types, and Processes

The syntax of usages and types are given by:

$$\begin{aligned} U \text{ (usages)} & ::= \mathbf{0} \mid ?U \mid !U \mid (U_1 \mid U_2) \mid U_1 + U_2 \mid X \mid \mu X.U \\ \tau \text{ (types)} & ::= \mathbf{chan}_U[\tau_1, \dots, \tau_n] \end{aligned}$$

The syntax of usages is almost the same as that of 2-label BPP, except that  $X$  may be unguarded in  $\mu X.U$ . For example,  $\mu X.X$  is allowed and is sometimes distinguished from  $\mathbf{0}$  [10,9]. The transition relation  $U \xrightarrow{l} U'$  (where  $l \in \{?, !\}$ ) is the same as that of  $\mathbf{BPP}_{\{?, !\}}$ . We often omit  $\mathbf{0}$  and just write  $!$  and  $?$  for  $! \mathbf{0}$  and  $? \mathbf{0}$ . We write  $FV(U)$  for the set of free variables in  $U$ .

Table 1 summarizes intuitive meaning of usages. For example, the usage  $? \mid !$  describes a channel that should be used once for input and once for output in parallel.

The type  $\mathbf{chan}_U[\tau_1, \dots, \tau_n]$ , abbreviated to  $\mathbf{chan}_U[\tilde{\tau}]$ , describes a channel that should be used for passing a tuple of channels of types  $\tau_1, \dots, \tau_n$ , and used

**Table 1.** Intuitive Meaning of Usages

|                |  |
|----------------|--|
| <b>0</b>       | not used at all  |
| $?U$           | used for input, and then according to $U$                |
| $!U$           | used for output, and then according to $U$               |
| $U_1 \mid U_2$ | used according to $U_1$ and $U_2$ , possibly in parallel |
| $U_1 + U_2$    | used according to either $U_1$ or $U_2$                  |
| $X$            | usage variable bound by $\mu$ .                          |
| $\mu X.U$      | used recursively according to $X = U$ .                  |

according to  $U$ . For example, the type  $\mathbf{chan}_{?!}[]$  describes a channel that should be first used for receiving, and then for sending a null tuple. A channel of type  $\mathbf{chan}_?[\mathbf{chan}_![]]$  should be used for receiving a channel, and then the received channel should be used for sending a null tuple.

The subtyping relation  $\tau_1 \leq \tau_2$  (which means that a value of type  $\tau$  may be used as a value of type  $\tau'$ ) is defined by:  $\mathbf{chan}_U[\tilde{\tau}] \leq \mathbf{chan}_{U'}[\tilde{\tau}]$  if and only if  $U \leq U'$ . Here, the *subusage* relation  $U \leq U'$  means that  $U$  represents a more liberal usage of channels, so that a channel of usage  $U$  may be used as a channel of usage  $U'$ . For example,  $?+! \leq ?$  should hold. There are several reasonable definitions of the subusage relation [7,8,9,10], depending on the property that should be ensured by the type system. The following definition is the simplest one among such reasonable definitions; other definitions are discussed later.

**Definition 7.**  $U_1 \leq U_2 \stackrel{def}{\iff} U_2 \leq_{tr} U_1$ .

Here,  $\leq_{tr}$  is the trace inclusion relation for  $\mathbf{BPP}_{\{?,!\}}$ .

The syntax of processes is given by:

$$P ::= \mathbf{0} \mid x![y_1, \dots, y_n].P \mid x?[y_1, \dots, y_n].P \mid (P \mid Q) \mid *P \mid (\nu x : U) P$$

A sequence  $y_1, \dots, y_n$  is abbreviated to  $\tilde{y}$ . The process  $x![\tilde{y}].P$  sends the tuple  $[\tilde{y}]$  of channels on channel  $x$ , and then behaves like  $P$ . The process  $x?[\tilde{y}].P$  waits to receive a tuple consisting of channels  $\tilde{z}$  on channel  $x$ , binds  $\tilde{y}$  to them, and then behaves like  $P$ . The process  $P \mid Q$  runs  $P$  and  $Q$  in parallel, and the process  $*P$  runs infinitely many copies of  $P$  in parallel. The process  $(\nu x : U) P$  creates a fresh communication channel, binds  $x$  to it, and then behaves like  $P$ . An important point here is that  $x$  is annotated with a usage  $U$ , which specifies a programmer's intention on how  $x$  should be used. As observed later, this usage declaration makes the type system described below undecidable. We do not consider choice  $P + Q$  and name matching  $[x = y]P$ ; The type system remains undecidable in the presence of those constructors.

*Example 1.* In the  $\pi$ -calculus, a lock (i.e., a binary semaphore) can be expressed as a channel, where the locked (unlocked, resp.) state is represented by the absence (presence, resp.) of a message. For example, the process  $lck?[].x?[y].lck![]$  locks  $lck$ , reads from  $x$ , and then releases  $lck$ . To enforce that the channel  $lck$  is indeed used as a lock (i.e., the channel is first used for output to initialize

the lock, and then used according to  $?!$  an arbitrary number of times), one can declare a usage of  $lck$  as  $(\nu lck : (! | \mu X. (\mathbf{0} + (?! | X)))) P$ . The type system introduced in the next subsection ensures that  $P$  uses  $lck$  according to the declared usage.

### 5.2 Type System

A type judgment for processes is of the form  $\Gamma \vdash P$ , where  $\Gamma$  is a type environment of the form  $x_1 : \tau_1, \dots, x_n : \tau_n$ . It means that  $P$  behaves as specified by  $\Gamma$ . For example,  $x : \mathbf{chan}_\tau[\mathbf{chan}_![]] \vdash P$  means that  $P$  uses  $x$  for receiving a channel of type  $\mathbf{chan}_![]$ , and then uses the received channel for sending a null tuple.

Typing rules and related definitions are given in Figure 2.

*Remark 1.* Although a usage of the form  $U_1 + U_2$  does not appear in Figure 2, it can be introduced by rule T-SUB. For example, the process:  $x![y] | x?[z]. z![] | x?[z]. z?[]$ .  $\mathbf{0}$  is typed under  $x : \mathbf{chan}_!|?|?[\mathbf{chan}_!+?[]], y : \mathbf{chan}_!+?[]$ .

#### Operation on type environments:

$$(\Gamma_1 | \Gamma_2)(x) = \begin{cases} (\Gamma_1(x)) | (\Gamma_2(x)) & \text{if } x \in \text{dom}(\Gamma_1) \cap \text{dom}(\Gamma_2) \\ \Gamma_1(x) & \text{if } x \in \text{dom}(\Gamma_1) \setminus \text{dom}(\Gamma_2) \\ \Gamma_2(x) & \text{if } x \in \text{dom}(\Gamma_2) \setminus \text{dom}(\Gamma_1) \end{cases}$$

where  $\mathbf{chan}_{U_1}[\tilde{\tau}] | \mathbf{chan}_{U_2}[\tilde{\tau}] = \mathbf{chan}_{U_1 | U_2}[\tilde{\tau}]$   
 $(*\Gamma)(x) = *( \Gamma(x) )$   
 where  $*\mathbf{chan}_U[\tilde{\tau}] = \mathbf{chan}_{\mu X.(U | X)}[\tilde{\tau}]$

#### Typing:

|  |          |   |          |
|--|----------|---|----------|
| $\frac{}{\emptyset \vdash \mathbf{0}}$   | (T-ZERO) | $\frac{\Gamma \vdash P}{*\Gamma \vdash *P}$   | (T-REP)  |
| $\frac{\Gamma \vdash P \quad \Delta \vdash Q}{\Gamma   \Delta \vdash P   Q}$   | (T-PAR)  | $\frac{\Gamma, x : \tau \vdash P \quad \tau' \leq \tau}{\Gamma, x : \tau' \vdash P}$  | (T-SUB)  |
| $\frac{\Gamma, x : \mathbf{chan}_U[\tilde{\tau}] \vdash P}{\Gamma \vdash (\nu x : U) P}$   | (T-NEW)  | $\frac{\Gamma \vdash P \quad U \leq \mathbf{0}}{\Gamma, x : \mathbf{chan}_U[\tilde{\tau}] \vdash P}$  | (T-WEAK) |
| $\frac{\Gamma, x : \mathbf{chan}_U[\tilde{\tau}] \vdash P}{(\Gamma, x : \mathbf{chan}_!U[\tilde{\tau}]   \tilde{y} : \tilde{\tau} \vdash x![\tilde{y}]. P)}$ | (T-OUT)  | $\frac{\Gamma, x : \mathbf{chan}_U[\tilde{\tau}], \tilde{y} : \tilde{\tau} \vdash P}{\Gamma, x : \mathbf{chan}_?U[\tilde{\tau}] \vdash x?[\tilde{y}]. P}$ | (T-IN)   |

Fig. 2. A Behavioral Type System

### 5.3 Undecidability of Type Checking Problem

We show that the problem of deciding whether  $\emptyset \vdash P$  holds or not is undecidable. The key observation for the proof is that, given two usages  $U_1$  and  $U_2$ , we can construct a process  $P$  such that  $\emptyset \vdash (\nu x : U_1) P$  if and only if  $U_1 \leq U_2$ . We use show the following key lemma.

**Lemma 6.** *Let  $U$  be a usage and suppose  $FV(U) \subseteq \{X_1, \dots, X_n\}$ . Then there exists a process  $P$  such that the followings are equivalent for any  $U', U_1, \dots, U_n$ .*

1.  $U' \leq [U_1/X_1, \dots, U_n/X_n]U$
2.  $x_1 : \mathbf{chan}_{U_\perp}[\mathbf{chan}_{U_1}[]], \dots, x_n : \mathbf{chan}_{U_\perp}[\mathbf{chan}_{U_n}[]], r : \mathbf{chan}_{U'}[] \vdash P$ .

Here,  $U_\perp = \mu X.(\mathbf{0} + ?X + !X)$ .

We obtain the following result as a corollary of Lemma 6 and Corollary 11.

**Theorem 3.** *The relation  $\emptyset \vdash P$  is undecidable.*

*Proof.* Let  $U_1, U_2$  be usages. By Lemma 6, there exists a process  $P_1$  such that  $r : \mathbf{chan}_{U_1}[] \vdash P_1$  if and only if  $U_1 \leq U_2$ . Hence,  $\emptyset \vdash (\nu r : U_1)P_1$  if and only if  $U_1 \leq U_2$ . Since the latter is undecidable, the type checking problem is also undecidable.  $\square$

*Undecidability results for other definitions of  $U_1 \leq U_2$*  The above undecidability result holds for various other definitions of the subusage relation. For example, let  $\leq \stackrel{def}{=} \geq_{sim}$ . Since Lemma 6 remains valid, and  $U_1 \leq_{sim} U_2$  is undecidable, the type checking problem is also undecidable. Here we sketch other definitions of subusage relations for which the type checking problem remains undecidable.

- Define a predicate  $U \downarrow$  inductively by the rules:

$$\frac{}{\mathbf{0} \downarrow} \quad \frac{U_1 \downarrow \quad U_2 \downarrow}{(U_1 | U_2) \downarrow} \quad \frac{U_i \downarrow}{(U_1 + U_2) \downarrow} \quad \frac{[\mu X.U/X]U \downarrow}{\mu X.U \downarrow}$$

Then add the condition  $U_1 \downarrow \Rightarrow U_2 \downarrow$  to the requirement for each element  $(U_1, U_2)$  in the simulation relation  $\leq_{sim}$ . Let  $\leq_{sim}^{ex}$  be the extended simulation relation, and define  $U_1 \leq U_2$  as  $U_2 \leq_{sim}^{ex} U_1$ .

- Extend the trace set using the above predicate:

$$extraces(U) = \{l_1 \dots l_n \mid U \xrightarrow{l_1} \dots \xrightarrow{l_n} U'\} \cup \{l_1 \dots l_n \downarrow \mid U \xrightarrow{l_1} \dots \xrightarrow{l_n} U' \downarrow\}$$

Then define  $U_1 \leq U_2$  as  $extraces(U_2) \subseteq extraces(U_1)$ .

- Add a transition  $U \xrightarrow{\tau} U'$  by introducing the synchronization rule:

$$\frac{U_1 \xrightarrow{?} U'_1 \quad U_2 \xrightarrow{!} U'_2}{U_1 | U_2 \xrightarrow{\tau} U'_1 | U'_2}$$

Then re-define the trace set, and define  $\leq$  as the trace inclusion relation.

*Remark 2.* The undecidability results above may be disappointing, given that behavioral type systems are useful for checking various properties [6,10,17,19,16] and that the above type system is one of the simplest forms of behavioral type systems. It should be noted, however, that the source of the undecidability result is the programmer’s capability to declare arbitrary usages (by  $(\nu x : U)$ ). In fact, Kobayashi’s type systems for deadlock-freedom and information flow [9,10] are much more complex, but the type checking problem is decidable (note that they do not allow type declaration). In order to allow declaration of usages as in this paper while keeping the decidability of type checking, we need to restrict the class of usages that can be declared by programmers. For example, type checking is decidable if the class of declared usages is restricted to the class of usages whose trace sets are deterministic Petri net languages [15].

## 6 Related Work

As already mentioned in Section 1, Hirshfeld [3] showed the undecidability of the trace equivalence for general BPP, and Hüttel [4] extended the result to show undecidability of other equivalence relations (except bisimilarity, which is decidable [2]). They [3,4] both encode Minsky machines into BPP. Since their encoding uses more than two action labels, their results do not immediately imply the undecidability for 2-label BPP.

Srba [5] proposed a general method for encoding a labeled transition system into a transition system with a *single* label, so that certain properties are preserved by the encoding. His encoding is, however, not applicable to BPP.

A number of behavioral type systems for the  $\pi$ -calculus have been proposed recently for checking various properties including deadlock, race, liveness, termination, and information flow [1,6,9,10,16,17]. Usage-based behavioral type systems studied in Section 5 were first proposed in [14] (in a less general form, without full recursion), and have been extended [1,9,10]. The undecidability result presented in this paper indicates that explicit usage or type declarations must be restricted in order to make those type systems decidable.

## 7 Conclusion

We have shown that the trace equivalence and simulation relation for 2-label BPP is undecidable. The undecidability result also implies the undecidability of certain behavioral type systems for the  $\pi$ -calculus.

**Acknowledgments.** We would like to thank Hans Hüttel for discussions on the undecidability of BPP equivalences. We would also like to thank anonymous reviewers for useful comments.

## References

1. Chaki, S., Rajamani, S., Rehof, J.: Types as models: Model checking message-passing programs. In: Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages, pp. 45–57. ACM Press, New York (2002)
2. Christensen, S., Hirshfeld, Y., Møller, F.: Decomposability, decidability and axiomatisability for bisimulation equivalence on basic parallel processes. In: Proceedings of IEEE Symposium on Logic in Computer Science, pp. 386–396. IEEE Computer Society Press, Los Alamitos (1993)
3. Hirshfeld, Y.: Petri nets and the equivalence problem. In: Meinke, K., Börger, E., Gurevich, Y. (eds.) CSL 1993. LNCS, vol. 832, pp. 165–174. Springer, Heidelberg (1994)
4. Hüttel, H.: Undecidable Equivalence for Basic Parallel Processes. In: Hagiya, M., Mitchell, J.C. (eds.) TACS 1994. LNCS, vol. 789, pp. 454–464. Springer, Heidelberg (1994)
5. Jirí, J.: On the power of labels in transition systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 277–291. Springer, Heidelberg (2001)

6. Kobayashi, N.: TyPiCal: A type-based static analyzer for the pi-calculus. Tool, available at <http://www.kb.ecei.tohoku.ac.jp/~koba/typical/>
7. Kobayashi, N.: A type system for lock-free processes. *Information and Computation* 177, 122–159 (2002)
8. Kobayashi, N.: Type systems for concurrent programs. In: Aichernig, B.K., Maibaum, T.S.E. (eds.) *Formal Methods at the Crossroads. From Panacea to Foundational Support*. LNCS, vol. 2757, pp. 439–453. Springer, Heidelberg (2003)
9. Kobayashi, N.: Type-based information flow analysis for the pi-calculus. *Acta Informatica* 42(4-5), 291–347 (2005)
10. Kobayashi, N.: A new type system for deadlock-free processes. In: Baier, C., Hermanns, H. (eds.) *CONCUR 2006*. LNCS, vol. 4137, pp. 233–247. Springer, Heidelberg (2006)
11. Kobayashi, N., Suto, T.: Undecidability of 2-label BPP equivalences and behavioural type systems for the  $\pi$ -calculus, Full version. Available (2007), from <http://www.kb.ecei.tohoku.ac.jp/~koba/publications.html>
12. Kobayashi, N., Suto, T.: Undecidability of BPP equivalences revisited, Available (2007), from <http://www.kb.ecei.tohoku.ac.jp/~koba/publications.html>
13. Milner, R.: *Communication and Concurrency*. Prentice-Hall, Englewood Cliffs (1989)
14. Sumii, E., Kobayashi, N.: A generalized deadlock-free process calculus. In: *Proc. of Workshop on High-Level Concurrent Language (HLCL'98)*. ENTCS, vol. 16(3), pp. 55–77 (1998)
15. Suto, T., Kobayashi, N.: Channel usage declaration for concurrent programming languages. *IPSJ Transaction on Programming* (to appear 2007) (in Japanese)
16. Yoshida, N.: Graph types for monadic mobile processes. In: Chandru, V., Vinay, V. (eds.) *Foundations of Software Technology and Theoretical Computer Science*. LNCS, vol. 1180, pp. 371–387. Springer, Heidelberg (1996)
17. Yoshida, N., Berger, M., Honda, K.: Strong normalisation in the pi-calculus. *Information and Computation* 191(2), 145–202 (2004)



# Ready Simulation for Concurrency: It's Logical!

Gerald Lüttgen<sup>1</sup> and Walter Vogler<sup>2</sup>

<sup>1</sup> Department of Computer Science, University of York, York YO10 5DD, U.K

luetttgen@cs.york.ac.uk

<sup>2</sup> Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany

vogler@informatik.uni-augsburg.de

**Abstract.** This paper provides new insight into the connection between the trace-based lower part of van Glabbeek's linear-time, branching-time spectrum and its simulation-based upper part. We establish that ready simulation is fully abstract with respect to failures inclusion, when adding the conjunction operator that was proposed by the authors in [TCS 373(1-2):19-40] to the standard setting of labelled transition systems with (CSP-style) parallel composition. More precisely, we actually prove a stronger result by considering a coarser relation than failures inclusion, namely a preorder that relates processes with respect to inconsistencies that may arise under conjunctive composition. Ready simulation is also shown to satisfy standard logic properties and thus commends itself for studying mixed operational and logic languages.

## 1 Introduction

Basic research in concurrency theory over the past 25 years has resulted in a wealth of process algebras [2, 8, 13] and temporal logics [4] for specifying and reasoning about concurrent processes. However, little research has been conducted on mixing process-algebraic and logic styles of specification in a single formalism. This is surprising since many popular software-engineering languages, including UML, permit such mixed specifications.

In [11] we proposed an approach to defining and reasoning about conjunction on labelled transition systems. Our setting consisted of standard labelled transition systems, augmented by an *inconsistency predicate* (cf. Sec. 2). While our conjunction operator is in essence a synchronous product on visible actions and an interleaving product on internal actions, the challenge was in dealing with inconsistencies. Inconsistencies may either arise when conjunctively composing two processes with different initial action sets (i.e., *ready sets*), or when a process has no other choice for some action than entering an inconsistent state. Our framework was equipped with *ready-tree semantics*, which is a variant of van Glabbeek's path-based possible-worlds semantics [6] that was inspired by Vegliani and De Nicola [17]. The resulting ready-tree preorder (for divergence-free consistent systems) turned out to be coarser than ready simulation and finer than failures inclusion and ready-trace inclusion, which implies the important feature that ready-tree semantics is sensitive to deadlock. We proved in [11] that

the ready-tree preorder is fully abstract under conjunction with respect to a naive *inconsistency preorder*<sup>1</sup> which allows an inconsistent specification only to be implemented by an inconsistent implementation.

This paper first shows that the ready-tree preorder is inadequate in the presence of concurrency, as it fails to be compositional for standard parallel composition, such as the parallel operator of CSP [8]. A different compositionality problem for the parallel composition of SCCS was already noted in [6]. We then establish our main result (cf. Sec. 3), namely that *ready simulation* [3], which adds to ordinary simulation the requirement that related processes must have identical ready sets, is fully abstract with respect to conjunction *and* parallel composition, for labelled transition systems with inconsistencies. Along the way, we adapt ready simulation to dealing with internal actions and inconsistencies. We also conduct several sanity checks: we verify that our conjunction operator indeed formalises *conjunction* regarding ready simulation, and prove further logic properties desired of ready simulation. Omitted proofs can be found in [12].

Our full-abstraction result provides an interesting insight into van Glabbeek's linear-time, branching-time spectrum [6], namely that conjunction on processes is a tool, via full abstraction, for relating the trace-based lower part of the spectrum to the simulation-based upper part. In addition, our results testify to the adequacy of ready simulation as the semantic basis for mixed process-algebraic and logic languages. Indeed, ready simulation eliminates the necessity for restrictions on the nesting of process-algebraic and logic constructs, such as the one employed by Olderog when embedding trace formulas into CSP [14].

## 2 Logic LTS, Conjunction and Parallel Composition

This section recalls the definitions of *Logic Labelled Transition Systems*, or Logic LTS for short, and the conjunction operator on Logic LTS which were introduced in [11]. It also lifts the parallel composition operator of CSP [8] to Logic LTS.

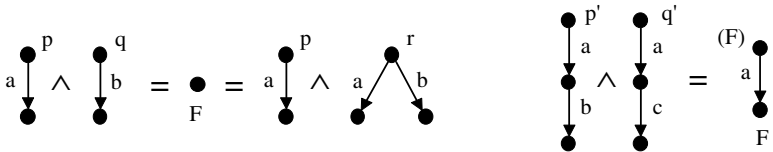


Fig. 1. Basic intuition behind conjunctive composition

Key to our setting is the consideration of *inconsistencies* which may arise under conjunctive composition. The idea is to mark a composed state between two processes as inconsistent, if one offers an action that the other cannot perform, i.e., if the processes have different *ready sets* [15]. Consider the processes  $p, q$

<sup>1</sup> i.e., the ready-tree preorder is the *coarsest precongruence* for conjunction which refines the inconsistency preorder.

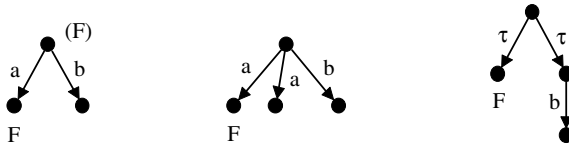


Fig. 2. Backward propagation of inconsistencies

and  $r$  of Fig. 1. Process  $p$  and  $q$  specify that exactly action  $a$  and respectively action  $b$  is offered initially, i.e., their ready sets are  $\{a\}$  and respectively  $\{b\}$ . Similarly, process  $r$  specifies that  $a$  and  $b$  are offered initially and thus has ready set  $\{a, b\}$ . Hence,  $p \wedge q$  as well as  $p \wedge r$  are *inconsistent* (or *false*) and should be tagged as such. Formally, our variant of LTS will be augmented by an *inconsistency predicate*, or *false-predicate*,  $F$ , so that  $p \wedge q, p \wedge r \in F$  in our example.

Inconsistency is a more tricky property, however, as it can propagate backwards along transitions. For example, in the conjunction  $p' \wedge q'$  shown in Fig. 1, both conjuncts require action  $a$  to be performed, whence  $p' \wedge q'$  should have an  $a$ -transition. But this transition does lead to an inconsistent state and, in the absence of any alternative  $a$ -transition leading to a consistent state,  $p' \wedge q'$  must itself be considered as inconsistent. In this spirit, inconsistency propagates backwards for the left process in Fig. 2, whereas it does not for the middle and right processes, as they can engage in an  $a$ -transition, respectively  $\tau$ -transition, that leads to a consistent state. As an aside, it is noted that the right process may be interpreted as a disjunction between the inconsistent process marked  $F$  which has empty behaviour, and the consistent process offering a  $b$ -transition.

*Logic Labelled Transition Systems.* Let  $\mathcal{A}$  be an alphabet with representatives  $a$  and  $b$ , and let  $\mathcal{A}_\tau$  denote  $\mathcal{A} \cup \{\tau\}$  with representatives  $\alpha$  and  $\beta$ . An LTS is a triple  $\langle P, \longrightarrow, F \rangle$  where  $P$  is the set of *processes* (states),  $\longrightarrow \subseteq P \times \mathcal{A}_\tau \times P$  is the *transition relation*, and  $F \subseteq P$  is the *inconsistency predicate*. We write (i)  $p \xrightarrow{\alpha} p'$  instead of  $\langle p, \alpha, p' \rangle \in \longrightarrow$ , (ii)  $p \xrightarrow{\alpha}$  instead of  $\exists p' \in P. p \xrightarrow{\alpha} p'$  and (iii)  $p \longrightarrow$  instead of  $\exists p' \in P, \alpha \in \mathcal{A}_\tau. p \xrightarrow{\alpha} p'$ . When  $p \xrightarrow{\alpha} p'$ , we say that process  $p$  can perform an  $\alpha$ -step to  $p'$ , and we call  $p'$  an  $\alpha$ -derivative. A process  $p$  that cannot engage in a  $\tau$ -transition, i.e.,  $p \not\xrightarrow{\tau}$ , is called *stable*. The *sort*  $\mathcal{A}_P$  of the LTS (and its processes) is the set of actions occurring in  $\longrightarrow$ .

We also require an LTS to satisfy the following  $\tau$ -purity condition:  $p \xrightarrow{\tau}$  implies  $\nexists a \in \mathcal{A}. p \xrightarrow{a}$ , for all  $p \in P$ . Hence, each process represents either an external or internal (disjunctive) choice between its outgoing transitions. This restriction reflects the fact that ready sets can only be observed at stable states, so that visible transitions leaving instable states are outside our observation. The LTSs of interest to us need to satisfy two further properties:

**Definition 1 (Logic LTS [11]).** An LTS  $\langle P, \longrightarrow, F \rangle$  is a *Logic LTS* if:

- (LTS1)  $F \subseteq P$  such that  $p \in F$  if  $\exists \alpha \in \mathcal{I}(p) \forall p' \in P. p \xrightarrow{\alpha} p'$  implies  $p' \in F$ ;
- (LTS2)  $p$  cannot stabilise implies  $p \in F$ .

<sup>2</sup> The additional, less relevant *true predicate* of [11] is omitted here for clarity.

The first condition formalises the backward propagation of inconsistencies as discussed above; here,  $\mathcal{I}(p)$  stands for the ready set  $\{\alpha \in \mathcal{A}_\tau \mid p \xrightarrow{\alpha}\}$  of process  $p$ . The second condition relates to *divergence*, i.e., infinite sequences of  $\tau$ -transitions, where divergence is viewed as catastrophic if a process cannot *stabilise*.

Before formalising our notion of stabilisation, we introduce several variants of weak transition relations which will prove useful in the sequel. We write  $p \xRightarrow{\epsilon} p'$  if  $p \xrightarrow{\tau}^* p'$  and  $p \xRightarrow{a} p'$  if  $\exists \bar{p}. p \xrightarrow{a} \bar{p} \xRightarrow{\epsilon} p'$ . Note that we do not consider  $\tau$ -transitions preceding a visible transition as we only need weak  $a$ -transitions originating from stable processes. If all processes along a computation  $p \xRightarrow{\epsilon} p'$  or  $p \xRightarrow{a} p'$ , including  $p$  and  $p'$ , are consistent, then we write  $p \xRightarrow{\epsilon}_F p'$  and  $p \xRightarrow{a}_F p'$ , respectively. If in addition,  $p'$  is stable, we write  $p \xRightarrow{\epsilon} \downarrow p'$  and  $p \xRightarrow{a} \downarrow p'$ , respectively. We may now define that a process  $p$  can stabilise if  $\exists p'. p \xRightarrow{\epsilon} \downarrow p'$ .

We will denote a transition  $p \xrightarrow{a} p'$  with  $p, p' \notin F$  by  $p \xrightarrow{a}_F p'$ . Moreover, whenever we mention a process  $p$  without stating a respective Logic LTS explicitly, we assume implicitly that such a Logic LTS  $\langle P, \longrightarrow, F \rangle$  is given. Finally, we let  $\text{ff}$  stand for the only process of the LTS  $\langle \{\text{ff}\}, \emptyset, \{\text{ff}\} \rangle$ , which represents the boolean constant *false*. Intuitively, any given process is either inconsistent, in which case it is equivalent to  $\text{ff}$ , or it is equivalent to a process from which no inconsistent process can be reached; the latter can simply be achieved by omitting inconsistent processes in LTSs and all transitions leading to them.

*Conjunction & parallel composition.* Our conjunction operator is a synchronous product for visible transitions and an asynchronous product for  $\tau$ -transitions, analogous to  $\parallel_{\mathcal{A}}$  defined below. However, we need to take care of inconsistencies. This is because, otherwise,  $p \wedge q$ , with  $p$  and  $q$  defined as in Fig. 11, would be a consistent process without any transitions.

**Definition 2 (Conjunction operator [11]).** The conjunction of two Logic LTSs  $\langle P, \longrightarrow_P, F_P \rangle$  and  $\langle Q, \longrightarrow_Q, F_Q \rangle$  is the Logic LTS  $\langle P \wedge Q, \longrightarrow_{P \wedge Q}, F_{P \wedge Q} \rangle$ :

- $P \wedge Q =_{\text{df}} \{p \wedge q \mid p \in P, q \in Q\}$
- $\longrightarrow_{P \wedge Q}$  is determined by the following operational rules:

$$\begin{array}{lll}
 p \xrightarrow{\tau}_P p' & \text{implies} & p \wedge q \xrightarrow{\tau}_{P \wedge Q} p' \wedge q \\
 q \xrightarrow{\tau}_Q q' & \text{implies} & p \wedge q \xrightarrow{\tau}_{P \wedge Q} p \wedge q' \\
 p \xrightarrow{a}_P p', q \xrightarrow{a}_Q q' & \text{implies} & p \wedge q \xrightarrow{a}_{P \wedge Q} p' \wedge q'
 \end{array}$$

- $F_{P \wedge Q}$  is the least set such that each  $p \wedge q \in F_{P \wedge Q}$  satisfies at least one of the following conditions:
  - (C1)  $p \in F_P$  or  $q \in F_Q$ ;
  - (C2)  $p \wedge q \xrightarrow{\tau}_{P \wedge Q}$  and  $\mathcal{I}(p) \neq \mathcal{I}(q)$ ;
  - (C3)  $\exists \alpha \in \mathcal{I}(p \wedge q) \forall p' \wedge q'. p \wedge q \xrightarrow{\alpha}_{P \wedge Q} p' \wedge q'$  implies  $p' \wedge q' \in F_{P \wedge Q}$ ;
  - (C4)  $p \wedge q$  cannot stabilise.

We are left with explaining Conds. (C1)–(C4). Firstly, a conjunction is inconsistent if a conjunct is inconsistent. Conds. (C2) and (C3) reflect our intuition of inconsistency and backward propagation. Cond. (C4) is added to enforce (LTS2).

**Definition 3 (Witness).** A *witness* is a set  $W \subseteq P \wedge Q$  such that, for all  $p \wedge q \in W$ , the following conditions hold:

- (W1)  $p, q \notin F$ ;
- (W2)  $p \xrightarrow{\tau}$  or  $q \xrightarrow{\tau}$  or  $\mathcal{I}(p) = \mathcal{I}(q)$ ;
- (W3)  $\forall \alpha \in \mathcal{A}_\tau. p \wedge q \xrightarrow{\alpha}$  implies  $\exists p' \wedge q' \in W. p \wedge q \xrightarrow{\alpha} p' \wedge q'$ ;
- (W4)  $p \wedge q$  can stabilise in  $W$ , i.e.,  $p \wedge q \xrightarrow{\tau} p_1 \wedge q_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} p_n \wedge q_n \xrightarrow{\tau} \dots$  with all  $p_i \wedge q_i \in W$ .

It is easy to check that the set of consistent processes  $\overline{F_{P \wedge Q}}$  of  $P \wedge Q$ , i.e., the complement of  $F_{P \wedge Q}$ , is a witness and is in fact the largest one in  $P \wedge Q$ . This implies the following straightforward proposition, giving us a useful tool for proving that the conjunction of two processes is consistent:

**Proposition 4.**  $p \wedge q \notin F_{P \wedge Q}$  if and only if  $\exists$  witness  $W. p \wedge q \in W$ .

For example, the concept of witness may be employed to prove the following properties of conjunctive composition:

- Lemma 5.**
1. If  $p \wedge q \xrightarrow{\tau} p' \wedge q' \notin F$  and  $p, q \notin F$ , then  $p \wedge q \notin F$ .
  2. Let  $p \xrightarrow{\epsilon} p', q \xrightarrow{\epsilon} q'$  and  $p' \wedge q' \notin F$ . Then,  $p \wedge q \xrightarrow{\epsilon} p' \wedge q'$ .

Finally, we adapt the parallel operator  $\parallel_A$  of CSP [8] to our setting, where  $A \subseteq \mathcal{A}$  denotes the *synchronisation alphabet*. Naturally, the parallel composition of two processes is inconsistent if either process is inconsistent.

**Definition 6 (Parallel operator).** The parallel composition of two Logic LTS  $\langle P, \longrightarrow_P, F_P \rangle$  and  $\langle Q, \longrightarrow_Q, F_Q \rangle$  for the synchronisation set  $A \subseteq \mathcal{A}$ , is the Logic LTS  $\langle P \parallel_A Q, \longrightarrow_{P \parallel_A Q}, F_{P \parallel_A Q} \rangle$ :

- $P \parallel_A Q =_{\text{df}} \{p \parallel_A q \mid p \in P, q \in Q\}$
- $\longrightarrow_{P \parallel_A Q}$  is determined by the following operational rules:
 

|  |         |   |
|--|---------|---|
| $p \xrightarrow{\alpha}_P p', \alpha \notin A, (\alpha = \tau \text{ or } q \not\xrightarrow{\tau}_Q)$ | implies | $p \parallel_A q \xrightarrow{\alpha}_{P \parallel_A Q} p' \parallel_A q$ |
| $q \xrightarrow{\alpha}_Q q', \alpha \notin A, (\alpha = \tau \text{ or } p \not\xrightarrow{\tau}_P)$ | implies | $p \parallel_A q \xrightarrow{\alpha}_{P \parallel_A Q} p \parallel_A q'$ |
| $p \xrightarrow{a}_P p', q \xrightarrow{a}_Q q', a \in A$  | implies | $p \parallel_A q \xrightarrow{a}_{P \parallel_A Q} p' \parallel_A q'$     |
- $p \parallel_A q \in F_{P \parallel_A Q}$  if  $p \in F_P$  or  $q \in F_Q$ .

Both conjunction and parallel composition are well-defined, i.e., the compositions of two Logic LTSs satisfy the conditions of Def. 1. In the sequel, we leave out indices of relations and predicates whenever the context is clear.

*Ready-tree semantics.* Our previous work [11] focused only on studying conjunction on Logic LTSs. It characterised the largest precongruence contained in the *inconsistency preorder*, which states that a consistent implementation  $p$  does never refine an inconsistent specification  $q$ <sup>3</sup>

**Definition 7 (Inconsistency preorder [11]).** The *inconsistency preorder*  $\sqsubseteq_F$  on processes is defined by  $p \sqsubseteq_F q$  if  $p \notin F$  implies  $q \notin F$ .

This definition agrees with the standard verification question whether an implementation satisfies its specification. When reading ‘satisfies’ logically as ‘implies’, it is clear that an inconsistent (i.e., ‘false’) specification can only be met by an inconsistent implementation.

Obviously, the inconsistency preorder is not compositional with respect to conjunction. Our characterisation of the fully-abstract preorder contained in  $\sqsubseteq_F$  and presented in [11] is based on a variant of the path-based possible-worlds semantics of [6, 17], to which we refer as *ready-tree semantics*. This semantics employs the notion of *observation tree*. An observation tree is a Logic LTS  $\langle V, \longrightarrow, \emptyset \rangle$  whose processes and transitions form a *deterministic tree* and whose processes (vertices) are stable; we refer to the tree’s root as  $v_0$ . We may now formalise our desired observations of a process  $p$ , called *ready trees*:

**Definition 8 (Ready tree [11]).** An observation tree  $v_0$  is a *ready tree of  $p$* , if there is a labelling  $h : V \rightarrow P$  satisfying the following conditions:

- (RT1)  $\forall v \in V. h(v)$  stable and  $h(v) \notin F$ ;
- (RT2)  $p \xrightarrow{\epsilon} h(v_0)$ ;
- (RT3)  $\forall v \in V, a \in \mathcal{A}. v \xrightarrow{a} v'$  implies  $h(v) \xrightarrow{a} h(v')$ ;
- (RT4)  $\forall v \in V. \mathcal{I}(v) = \mathcal{I}(h(v))$ .

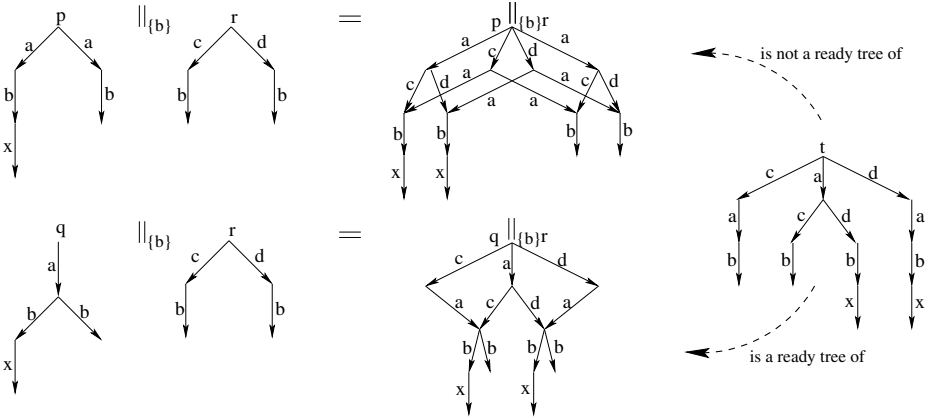
Intuitively, nodes  $v$  in a ready tree represent stable states  $h(v)$  of  $p$  (cf. the first part of Cond. (RT1)) and transitions represent stable, consistent computations (cf. Cond. (RT3)). Since such computations do not contain inconsistent states, no represented state must be in  $F$  (cf. the second part of Cond. (RT1)). Since  $p$  might not be stable, the root  $v_0$  of a ready tree represents a stable process reachable from  $p$  via some internal computation (cf. Cond. (RT2)). Moreover,  $v$  must mimic the ready set of  $h(v)$  (cf. Cond. (RT4)). In the following, we write  $\text{RT}(p)$  for the set of all ready trees of  $p$ ; note that  $\text{ff}$  has no ready tree.

**Definition 9 (Ready-tree preorder [11]).** The *ready-tree preorder*  $\subseteq_{\text{RT}}$  on processes is defined as ready-tree inclusion, i.e.,  $p \subseteq_{\text{RT}} q$  if  $\text{RT}(p) \subseteq \text{RT}(q)$ .

**Theorem 10 (Full-abstraction wrt. conjunction [11]).**  $\subseteq_{\text{RT}}$  is the largest precongruence in  $\sqsubseteq_F$ , when considering conjunction as the only operator.

---

<sup>3</sup> The reader familiar with [11] should note that we now write the implementation to the left and the specification to the right of the preorder symbol, in order to be consistent with the notational conventions of simulation-based preorders.



**Fig. 3.** Ready-tree semantics is not compositional for parallel composition

Unfortunately,  $\subseteq_{RT}$  is *not* a precongruence for parallel composition  $\parallel_A$ , which makes the preorder unsuitable for reasoning about concurrency. To see this, consider the Logic LTSs  $p$ ,  $q$  and  $r$  of Fig. 3. Here,  $p$  and  $q$  have the same ready trees, but  $t$  is a ready tree of  $q \parallel_{\{b\}} r$  but not of  $p \parallel_{\{b\}} r$ .

### 3 Full Abstraction Via Ready Simulation

We now establish our full-abstraction result of ready simulation wrt. the inconsistency preorder, when considering both conjunction and parallel composition.

**Definition 11 (Ready simulation on Logic LTS).** Let  $\langle P, \longrightarrow_P, F_P \rangle$  and  $\langle Q, \longrightarrow_Q, F_Q \rangle$  be two Logic LTS. A relation  $\mathcal{R} \subseteq P \times Q$  is a *stable ready simulation relation*, if the following conditions hold, for any  $\langle p, q \rangle \in \mathcal{R}$  and  $a \in \mathcal{A}$ :

- (RS1)  $p, q$  stable;
- (RS2)  $p \notin F_P$  implies  $q \notin F_Q$ ;
- (RS3)  $p \xrightarrow{a} p'$  implies  $\exists q'. q \xrightarrow{a} q'$  and  $\langle p', q' \rangle \in \mathcal{R}$ ;
- (RS4)  $p \notin F$  implies  $\mathcal{I}(p) = \mathcal{I}(q)$ .

We say that  $p$  is *stable ready simulated* by  $q$ , in symbols  $p \sqsubseteq_{RS} q$ , if there exists a stable ready simulation relation  $\mathcal{R}$  with  $\langle p, q \rangle \in \mathcal{R}$ . Further,  $p$  is *ready simulated* by  $q$ , written  $p \sqsubseteq_{RS} q$ , if  $\forall p'. p \xrightarrow{\epsilon} p'$  implies  $\exists q'. q \xrightarrow{\epsilon} q'$  and  $p' \sqsubseteq_{RS} q'$ . We write  $\approx_{RS}$  and  $=_{RS}$  for the kernel of  $\sqsubseteq_{RS}$  and  $\sqsubseteq_{RS}$ , respectively.

It is easy to see that  $\sqsubseteq_{RS}$  and  $\sqsubseteq_{RS}$  are preorders, and that  $p \sqsubseteq_{RS} q$  trivially holds if  $p \in F$ . Moreover, ready simulation  $\sqsubseteq_{RS}$  is contained in the ready-tree preorder  $\subseteq_{RT}$ , as essentially stated in [6], and conjunction and parallel composition are associative and commutative with respect to  $=_{RS}$ . Note that there are several ways how to define ready simulation [3, 6] for settings with internal actions [5]. Our variant is an adaptation of Glabbeek’s *stability respecting*

ready simulation may preorder to Logic LTS. Observe that replacing the premise  $p \xrightarrow{a} p'$  of (RS3) by  $p \xrightarrow{a}_F p'$  results in a finer preorder, unlike for some other simulation-based behavioural relations [13].

**Theorem 12 (Compositionality)**

1. Let  $p \xrightarrow{\sim}_{RS} q$ ,  $r$  be stable and  $A \subseteq \mathcal{A}$ . Then, (a)  $p \parallel_A r \xrightarrow{\sim}_{RS} q \parallel_A r$  as well as (b)  $p \wedge r \xrightarrow{\sim}_{RS} q \wedge r$ .
2. Let  $p \sqsubseteq_{RS} q$ ,  $r$  be an arbitrary process and  $A \subseteq \mathcal{A}$ . Then, (a)  $p \parallel_A r \sqsubseteq_{RS} q \parallel_A r$  and (b)  $p \wedge r \sqsubseteq_{RS} q \wedge r$ .

Regarding the proof, we only want to point out that it employs the proof tool of witness in order to reason about the consistency of conjunctively composed processes in Part (II). The following witness turns out to be sufficient for our purpose:

**Lemma 13.** *The set  $W =_{df} W_1 \cup W_2$  is a witness, where*

$$W_1 =_{df} \{q \wedge r \mid \exists p. p \xrightarrow{\sim}_{RS} q \text{ and } p \wedge r \notin F\};$$

$$W_2 =_{df} \{\bar{q} \wedge \bar{r} \mid \exists p, q, r, p', r', q', a. p \xrightarrow{\sim}_{RS} q, p \wedge r \notin F, p \wedge r \xrightarrow{a} p' \wedge r', p' \xrightarrow{\sim}_{RS} q', \\ \text{and } q \xrightarrow{a}_F \bar{q} \xrightarrow{\epsilon_2} q' \text{ and } r \xrightarrow{a}_F \bar{r} \xrightarrow{\epsilon_1} r' \text{ with } \{\epsilon_1, \epsilon_2\} = \{\epsilon, \tau\}\}.$$

*Full-abstraction result.* To prove our main result we encode the full behaviour of a stable process  $p$  into a single ready tree. The idea is to unfold  $p$  to a tree and to eliminate any nondeterminism by placing fresh actions into the tree.

**Definition 14 (Characteristic ready tree & context).** Let  $p$  be a process with Logic LTS  $\langle P, \longrightarrow, F \rangle$  having sort  $\mathcal{A}_P$ , let  $q$  be a process with sort  $\mathcal{A}_Q$ , and let  $p \xrightarrow{\epsilon} p_0$ .

1. The *characteristic ready tree*  $P_0$  of  $p$  with respect to  $p_0$  and  $q$  is a Logic LTS whose states are paths  $\pi \in P \times (\mathcal{A}_P \times P)^*$  of  $p$  originating in  $p_0$ , as well as such paths concatenated with *selection sets*  $D$  which are subsets of  $\mathcal{A}_P \times P$ . Formally, the state set  $P_0$  and transition relation  $\longrightarrow_{P_0}$  are inductively defined as follows, where  $\text{last}(\pi)$  denotes the last process on path  $\pi$  and the  $x_D \notin \mathcal{A}_P \cup \mathcal{A}_Q$  are fresh actions with respect to  $p$  and  $q$ :
  - $p_0 \in P_0$ ;
  - $\pi \xrightarrow{x_D}_{P_0} \pi D$  and  $\pi D \in P_0$ , if  $\pi \in P_0$ ,  $\forall \langle a, p \rangle \in D. \text{last}(\pi) \xrightarrow{a} p$  in  $P$  and  $\forall a \in \mathcal{I}(\text{last}(\pi)) \exists_1 \langle a, p \rangle \in D$ ;
  - $\pi D \xrightarrow{a}_{P_0} \pi a p$  and  $\pi a p \in P_0$ , if  $\pi D \in P_0$  and  $\langle a, p \rangle \in D$ .

We will write  $\langle p_0 \rangle$  instead of  $p_0$  when we wish to highlight that not the process  $p_0$  is meant, but the path consisting only of  $p_0$ .

2. The *characteristic context*  $K$  of  $p$  with respect to  $p_0$  and  $q$  is defined as the Logic LTS  $P_0$  augmented with the fresh process  $0$  and transitions
  - $\pi D \xrightarrow{a}_K 0$ , if  $\pi D \in P_0$ ,  $a \in \mathcal{A}_Q$  and  $\nexists p. \langle a, p \rangle \in D$ .



**Proposition 15.** *Let  $P_0$  be the characteristic ready tree of a process  $p$  wrt. some  $p_0$  and  $q$ , and let  $K$  be the respective characteristic context of  $p$ . Then,  $P_0$  is a ready tree of  $p \parallel_A \langle p_0 \rangle$ , where  $A =_{df} \mathcal{A}_P \cup \mathcal{A}_Q$  and  $\langle p_0 \rangle$  is the root of  $K$ .*

*Proof.*  $P_0$  is an observation tree by construction, since it is a deterministic tree and since all its vertices are stable processes. We define a mapping  $h_0$  from the vertices in  $P_0$  to processes in  $P \parallel_A K$  by  $h_0(\pi) =_{df} \text{last}(\pi) \parallel_A \pi$  and  $h_0(\pi D) =_{df} \text{last}(\pi) \parallel_A \pi D$ , and verify Conds. (RT1)–(RT4) of Def. 8.

(RT1) This is trivial since  $\text{last}(\pi)$ ,  $\pi$  and  $\pi D$  are all stable and not in  $F$ .

(RT2) We have  $p \parallel_A \langle p_0 \rangle \xrightarrow{\epsilon} p_0 \parallel_A \langle p_0 \rangle$  by construction.

(RT3) If  $\pi \xrightarrow{x_D}_{P_0} \pi D$ , then  $\pi \xrightarrow{x_D}_K \pi D$  by construction of  $K$ . Since  $x_D$  is a “fresh” action,  $h_0(\pi) = \text{last}(\pi) \parallel_A \pi \xrightarrow{x_D}_F \text{last}(\pi) \parallel_A \pi D = h_0(\pi D)$ . If  $\pi D \xrightarrow{a}_{P_0} \pi a p$ , then  $\text{last}(\pi) \xrightarrow{a} p$  and  $\pi D \xrightarrow{a}_K \pi a p$  by the construction of  $K$ . As  $a \in A$ , we have  $h_0(\pi D) = \text{last}(\pi) \parallel_A \pi D \xrightarrow{a} p \parallel_A \pi a p = h_0(\pi a p)$ .

(RT4) Observe that the ready set of state  $\pi D$  in  $K$  is the initial action set  $\mathcal{I}(\text{last}(\pi))$  of the last process of path  $\pi$  in  $P$  plus all actions in  $\mathcal{A}_Q$ , whereas the same state in  $P_0$  has only ready set  $\mathcal{I}(\text{last}(\pi))$ . By the operational rules for parallel composition we obtain:

- $\mathcal{I}_{P \parallel_A K}(\text{last}(\pi) \parallel_A \pi) = (\mathcal{I}_P(\text{last}(\pi)) \cap \mathcal{I}_K(\pi) \cap A) \cup (\mathcal{I}_P(\text{last}(\pi)) \setminus A) \cup (\mathcal{I}_K(\pi) \setminus A) = \emptyset \cup \emptyset \cup \mathcal{I}_K(\pi) = \mathcal{I}_{P_0}(\pi)$ .
- $\mathcal{I}_{P \parallel_A K}(\text{last}(\pi) \parallel_A \pi D) = (\mathcal{I}_P(\text{last}(\pi)) \cap \mathcal{I}_K(\pi D) \cap A) \cup (\mathcal{I}_P(\text{last}(\pi)) \setminus A) \cup (\mathcal{I}_K(\pi D) \setminus A) = (\mathcal{I}_P(\text{last}(\pi)) \cap (\mathcal{I}_P(\text{last}(\pi)) \cup \mathcal{A}_Q) \cap A) \cup \emptyset \cup ((\mathcal{I}_P(\text{last}(\pi)) \cup \mathcal{A}_Q) \setminus A) = \mathcal{I}_P(\text{last}(\pi)) = \mathcal{I}_{P_0}(\pi D)$ ; note that the last equality is due to the construction of  $P_0$  from  $P$ .  $\square$

Observe that  $P_0$  is not a ready tree of  $p$  itself due to the fresh actions inserted in  $P_0$ ; these actions are added to  $p$  via the parallel context  $K$ . Together, characteristic ready trees and Prop. 15 are the key for proving our main result:

**Theorem 16 (Full abstraction).** *The largest precongruence contained in  $\sqsubseteq_F$ , with respect to parallel composition and conjunction, equals  $\sqsubseteq_{RS}$ .*

*Proof.* Because of Thm. 12 and Thm. 10 [11], as well as the fact that ready simulation is contained in the ready-tree preorder  $\subseteq_{RT}$  and thus in  $\sqsubseteq_F$  [11], it is sufficient to prove that  $\sqsubseteq_{RS}$  subsumes the largest precongruence  $\subseteq_{RT}^+$  contained in  $\subseteq_{RT}$ . Consider processes  $p$  and  $q$  with Logic LTSs  $P$  and  $Q$  and sorts  $\mathcal{A}_P$  and  $\mathcal{A}_Q$ . We let  $\mathcal{A}_{PQ}$  stand for  $\mathcal{A}_P \cup \mathcal{A}_Q$ , and abbreviate  $\parallel_{\mathcal{A}_{PQ}}$  by  $\parallel$ .

Now assume  $p \subseteq_{RT}^+ q$ , and consider some  $p_0$  such that  $p \xrightarrow{\epsilon} p_0$ . Because of  $p \subseteq_{RT}^+ q$  and Prop. 15, we have  $P_0 \in \text{RT}(q \parallel \langle p_0 \rangle)$  due to some mapping  $h$  (cf. Def. 8); in particular,  $q \notin F$ . Here,  $P_0$  is the characteristic ready tree of  $p$  with respect to  $p_0$  and  $q$ . To prove our claim, it is sufficient to establish that

$$\mathcal{R}_0 =_{df} \{ \langle p', q' \rangle \mid \exists \pi. \text{last}(\pi) = p' \text{ and } h(\pi) = q' \parallel \pi \}$$

is a stable ready simulation relation. Thus, let  $\langle p', q' \rangle \in \mathcal{R}_0$  due to  $\pi$ .

- (RS1)  $h(\pi)$  is stable, whence  $q'$  is. Moreover,  $\text{last}(\pi)$  is stable by construction.
- (RS2)  $h(\pi) \notin F$  implies  $q' \notin F$ .
- (RS3) Let  $p' \xrightarrow{a} p''$  and  $\pi \xrightarrow{x_D} \pi D$  with  $\langle a, p'' \rangle \in D$  for some  $p''$ . Then,  $\pi D \xrightarrow{a} \pi a p''$ . Moreover,  $h(\pi D) = q' \parallel \pi D$ , whence  $q' \parallel \pi D \xrightarrow{a} h(\pi a p'') = q'' \parallel \pi a p''$  for some  $q''$  by (RT3), as well as  $q' \xrightarrow{a} q''$  and  $\langle p'', q'' \rangle \in \mathcal{R}_0$  due to  $\pi a p''$ .
- (RS4) We have  $p' \notin F$  by construction. Choose some  $D$  with  $\pi \xrightarrow{x_D} \pi D$ , whence  $h(\pi D) = q' \parallel \pi D$ . Now,  $\mathcal{I}(p') = \mathcal{I}(\pi D)$  in  $P_0$  by construction of  $P_0$ . The latter equals  $\mathcal{I}(q' \parallel \pi D)$  by (RT4), which in turn equals the set  $\mathcal{I}(q')$  since  $\mathcal{A}_Q \subseteq \mathcal{I}(\pi D) \subseteq \mathcal{A}_{PQ}$ , for  $\mathcal{I}(\pi D)$  in the characteristic context. Hence,  $\mathcal{I}(p') = \mathcal{I}(q')$ .

Thus,  $\mathcal{R}_0$  is a stable ready simulation relation. Finally observe  $h(p_0) = q_0 \parallel \langle p_0 \rangle$  for some  $q_0$  such that  $q \parallel \langle p_0 \rangle \xrightarrow{\epsilon} q_0 \parallel \langle p_0 \rangle$  (by (RT2)); therefore,  $q \xrightarrow{\epsilon} q_0$  and  $\langle p_0, q_0 \rangle \in \mathcal{R}_0$  due to  $\langle p_0 \rangle$ .

Summarising, we have shown that, for each  $p_0$  with  $p \xrightarrow{\epsilon} p_0$ , there exists some  $q_0$  satisfying  $q \xrightarrow{\epsilon} q_0$  and  $p_0 \sqsubseteq_{RS} q_0$ . Hence,  $p \sqsubseteq_{RS} q$ . □

One way to guarantee the existence of the fresh actions required in the construction of characteristic ready trees is to assume an uncountable alphabet  $\mathcal{A}$  and to restrict ourselves to those processes that are finitely branching with respect to  $\xrightarrow{a}$ , for all  $a \in \mathcal{A}$ , and have a countable sort. Then, context  $K$  and the characteristic ready trees are also finitely branching and have countable sorts.

*Logic properties of ready simulation.* We conclude this section by highlighting some logic properties of ready simulation.

**Theorem 17 ( $\wedge$  is And).** (1)  $r \sqsim_{RS} p \wedge q$  if and only if  $r \sqsim_{RS} p$  and  $r \sqsim_{RS} q$ ;  
 (2)  $r \sqsubseteq_{RS} p \wedge q$  if and only if  $r \sqsubseteq_{RS} p$  and  $r \sqsubseteq_{RS} q$ .

As for the compositionality proof of ready simulation wrt. conjunction, the proof of this theorem uses the concept of witness for reasoning about inconsistencies:

**Lemma 18.** *The set  $W' =_{df} W'_1 \cup W'_2$  is a witness, where*  
 $W'_1 =_{df} \{p \wedge q \mid \exists r. r \sqsim_{RS} p, r \sqsim_{RS} q \text{ and } r \notin F\}$   
 $W'_2 =_{df} \{\bar{p} \wedge \bar{q} \mid \exists r, p, q, r', p', q', a. r \sqsim_{RS} p, r \sqsim_{RS} q, r \xrightarrow{a} r', p \xrightarrow{a}_F \bar{p} \xrightarrow{\epsilon_1} p' \text{ and } q \xrightarrow{a}_F \bar{q} \xrightarrow{\epsilon_2} q' \text{ with } \{\epsilon_1, \epsilon_2\} = \{\epsilon, \tau\}, r' \sqsim_{RS} p' \text{ and } r' \sqsim_{RS} q'\}.$

Conjunction also satisfies further standard logic properties:

**Proposition 19 (Logic properties of ready simulation)**

1.  $p \wedge \text{ff} =_{RS} \text{ff}$ ;  $p \wedge \text{ff} \approx_{RS} \text{ff}$  if  $p$  stable;
2.  $p \wedge q \sqsubseteq_{RS} p$ ;  $p \wedge q \sqsim_{RS} p$  if  $p, q$  stable;
3.  $p \wedge p =_{RS} p$ ;
4.  $p \wedge q =_{RS} p$  if and only if  $p \sqsubseteq_{RS} q$ .

In our previous work we also considered a disjunction operator  $\vee$  on Logic LTSs. This operator was defined as internal choice, i.e.,  $p \vee q$  can perform an internal  $\tau$ -transition to both  $p$  and  $q$ , where  $p \vee q$  is considered to be inconsistent if

both  $p$  and  $q$  are. Due to space constraints we do not include disjunction here, but simply note that ready simulation is compositional for disjunction and that the dual properties to the ones of Prop. 19 hold. The validity of these statements is not difficult to check. Moreover, the distributivity laws hold, too.

## 4 Related Work

This section briefly discusses related work; a full discussion can be found in [11]. Firstly, our ready-tree semantics is in essence the *path-based possible-worlds semantics* of van Glabbeek [6] which goes back to Vegliani and De Nicola [17], and our ready simulation was first suggested by Bloom et al. [3]. However, in contrast to the standard notions of these semantics, our setting deals with internal actions as well as inconsistencies.

Traditional research has often avoided explicitly mixing operational and logic styles of specification by translating one style into the other. Operational content may be translated into logic formulas, as is implicitly done in [7, 10], where logic implication serves as refinement relation [1]. Dually, logic content may be translated into operational content. This is the case in automata-theoretic work, such as in Kurshan’s work on  $\omega$ -automata [9], which includes synchronous and asynchronous composition operators and uses maximal trace inclusion as refinement relation. However, both logic implication and trace inclusion are insensitive to deadlock and are thus inadequate in the presence of concurrency.

A seminal approach to compositional refinement in a mixed setting was proposed by Olderog in [14], where process-algebraic constructs are combined with trace formulas expressed in a predicate logic and where failure semantics forms the semantic basis of refinement. In this approach, trace formulas can serve as processes, but not vice versa. Thus, and in contrast to our present work, freely mixing operational and logic specification styles is not supported and, in particular, conjunction cannot be applied to processes.

Finally, it should be noted that the term *consistency* as used here is different from the one in [16], where two specifications are defined as consistent if they have at least one implementation in common. In our setting, a process  $p \notin F$  is called consistent, while  $p \wedge q$  implements both  $p$  and  $q$ , for arbitrary  $p, q$ . Thm. 17 also implies that  $p$  and  $q$  are consistent in the sense of [16], if  $p \wedge q \notin F$  in our setting.

## 5 Conclusions and Future Work

This paper proved that ready simulation [3] is fully abstract with respect to conjunction and parallel composition on Logic LTS. In this sense, ready simulation is indeed a “logical” semantics. Establishing this result was non-trivial due to the challenges that arise when dealing with inconsistencies under conjunctive composition. This is evidenced by the complex compositionality proof with respect to conjunction, as well as the two-step “largest” precongruence proof that relied on our previous full-abstraction work on ready-tree semantics [11].

Our results show that conjunction is a tool for relating trace-based semantics to simulation-based semantics, via the concept of full abstraction. This sheds additional light onto van Glabbeek's linear-time, branching-time spectrum [6]. Moreover, our results imply that ready simulation commends itself as a suitable behavioural relation for reasoning about specifications given in a mixed operational and logic style. Indeed, future work shall employ ready simulation within novel algebras that will combine process-algebraic and temporal-logic operators.

*Acknowledgements.* We wish to thank the anonymous referees for their insightful and constructive comments. The first author also acknowledges support by the EPSRC under grant no. EP/E034853/1.

## References

- [1] Abadi, M., Plotkin, G.D.: A logical view of composition. TCS 114(1), 3–30 (1993)
- [2] Bergstra, J.A., Ponse, A., Smolka, S.A.: Handbook of Process Algebra. Elsevier, Amsterdam (2001)
- [3] Bloom, B., Istrail, S., Meyer, A.R.: Bisimulation can't be traced. J. ACM 42(1), 232–268 (1995)
- [4] Emerson, E.A.: Temporal and modal logic. In: Handbook of Theoretical Computer Science, vol. B, pp. 995–1072. North-Holland, Amsterdam (1990)
- [5] van Glabbeek, R.: The linear time – branching time spectrum II (1993) Available at <http://theory.stanford.edu/~rvg/abstracts.html#26>
- [6] van Glabbeek, R.: The linear time – branching time spectrum I. In: Handbook of Process Algebra, ch. 1, pp. 3–99. Elsevier, Amsterdam (2001)
- [7] Graf, S., Sifakis, J.: A logic for the description of non-deterministic programs and their properties. Inform. & Control 68(1–3), 254–270 (1986)
- [8] Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs (1985)
- [9] Kurshan, R.P.: Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach. Princeton Univ. Press, Princeton (1994)
- [10] Lamport, L.: The temporal logic of actions. TOPLAS 16(3), 872–923 (1994)
- [11] Lüttgen, G., Vogler, W.: Conjunction on processes: Full-abstraction via ready-tree semantics. TCS 373(1–2), 19–40 (2007)
- [12] Lüttgen, G., Vogler, W.: Ready simulation for concurrency: It's logical! Techn. rep. 2007-4, Inst. f. Informatik, Univ. Augsburg (2007) <http://www.informatik.uni-augsburg.de/forschung/reports/>
- [13] Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
- [14] Olderog, E.-R.: Nets, Terms and Formulas. Cambridge Tracts in Theoretical Computer Science, vol. 23. Cambridge Univ. Press, Cambridge (1991)
- [15] Olderog, E.-R., Hoare, C.A.R.: Specification-oriented semantics for communicating processes. Acta Informatica 23(1), 9–66 (1986)
- [16] Steen, M., Derrick, J., Boiten, E., Bowman, H.: Consistency of partial process specifications. In: Haerberer, A.M. (ed.) AMAST 1998. LNCS, vol. 1548, pp. 248–262. Springer, Heidelberg (1998)
- [17] Vegliani, S., De Nicola, R.: Possible worlds for process algebras. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 179–193. Springer, Heidelberg (1998)

# Continuous Capacities on Continuous State Spaces<sup>\*</sup>

Jean Goubault-Larrecq

LSV, ENS Cachan, CNRS, INRIA Futurs  
61, avenue du président-Wilson, F-94235 Cachan  
[goubault@lsv.ens-cachan.fr](mailto:goubault@lsv.ens-cachan.fr)

**Abstract.** We propose axiomatizing some stochastic games, in a continuous state space setting, using continuous belief functions, resp. plausibilities, instead of measures. Then, stochastic games are just variations on continuous Markov chains. We argue that drawing at random along a belief function is the same as letting the probabilistic player P play first, then letting the non-deterministic player C play demonically. The same holds for an angelic C, using plausibilities instead. We then define a simple modal logic, and characterize simulation in terms of formulae of this logic. Finally, we show that (discounted) payoffs are defined and unique, where in the demonic case, P maximizes payoff, while C minimizes it.

## 1 Introduction

Consider Markov chains: these are transition systems, which evolve from state  $x \in X$  by drawing the next state  $y$  in the state space  $X$  according to some probability distribution  $\theta(x)$ . One may enrich this model to take into account decisions made by a *player* P, which can take actions  $\ell$  in some set  $L$ . In state  $x \in X$ , P chooses an action  $\ell \in L$ , and draws the next state  $y$  according to a probability distribution  $\theta_\ell(x)$  depending on  $\ell \in L$ : these are *labeled Markov processes* (LMPs) [8]. Adding rewards  $r_\ell(x)$  on taking action  $\ell$  from state  $x$  yields *Markov decision processes* [11]. The main topic there is to evaluate strategies that maximize the expected payoff, possibly discounted.

These notions have been generalized in many directions. Consider stochastic games, where there is not one but several players, with different goals. In security protocols, notably, it is meaningful to assume that the honest agents collectively define a player P as above, who may play probabilistically, and that attackers define a second player C, who plays *non-deterministically*. Instead of drawing the next state at random, C deliberately chooses its next state, typically to minimize P's expected payoff or to maximize the probability that a bad state is reached—this is *demonic* non-determinism.

A nice idea of F. Lavolette and J. Desharnais (private comm., 2003), which we develop, is that the theory of these games could be simplified by relaxing the requirements of Markov chains: if  $\nu = \theta_\ell(x)$  is not required to be a measure, but the additivity requirement is relaxed to sub-additivity (i.e.,  $\nu(A) + \nu(B) \leq \nu(A \cup B)$  for disjoint measurable sets  $A, B$ ), then such “preprobabilities” include both ordinary probabilities and the following funny-looking *unanimity game*  $u_A$ , which represents the demonic non-deterministic choice of an element from the set  $A$ : the preprobability  $u_A(B)$  of drawing

---

<sup>\*</sup> Partially supported by the INRIA ARC ProNoBis.

an element in  $B$  is 1 if  $A \subseteq B$ , 0 otherwise. The intuition is as follows. Assume that, starting from state  $x$ , you would like the next state  $y$  to be in  $B$ . A demonic adversary  $C$  will then strive to pick  $y$  outside  $B$ . Now if  $C$ 's moves are given by  $\delta_\ell(x) = u_A$ , then either  $A \not\subseteq B$ , then it is  $C$ 's interest to pick  $y$  from  $A \setminus B$ , so that the preprobability that  $y$  be in  $B$  is 0; or  $A \subseteq B$ , then  $C$  is forced to play  $y \in B$ , and the preprobability is 1.

However, sub-additive set functions are not quite the right notion; and second (which does not detract from F. Lavolette and J. Desharnais' great intuition), the right notions had been invented by economists in the 1950s under the name of "cooperative game with transferable utility" [22] and by statisticians in the 1960s under the names of belief functions and plausibilities, while capacities and Choquet integration are even more ancient [4]. A nice survey is [13]. These notions are well-known in discrete state spaces. Our generalization to topological spaces is new, and non-trivial. The spaces we consider include finite spaces as well as infinite ones such as  $\mathbb{R}^n$ , but also cpos and domains.

**Outline.** We introduce necessary mathematical notions in Section 2. We then develop the theory of continuous games, and continuous belief functions in particular in Section 3, showing in a precise sense how the latter model both probabilistic and demonic non-deterministic choice. We then recall the Choquet integral in Section 4, and show how taking averages reflects the fact that  $C$  aims at minimizing  $P$ 's gains. We briefly touch the dual notion of plausibilities (angelic non-determinism) in passing. Finally, we define ludic transition systems, the analogue of Markov chains, except using continuous games, in Section 5, and define a notion of simulation topologies. We show that the coarsest simulation topology is exactly that defined by a simple modal logic, à la Larsen-Skou [19]. This illustrates how continuous games allow us to think of certain stochastic games as really being just LMPs, only with a relaxed notion of probability.

This work is a summary of most of Chapters 1-9 of [14], in which all proofs, and many more results can be found.

**Related Work.** Many models of Markov chains or processes, or stochastic games are discrete or even finite-state. Desharnais *et al.* [8] consider LMPs over *analytic* spaces, a class of topological spaces that includes not only finite spaces but also spaces such as  $\mathbb{R}^n$ . They show an extension of Larsen and Skou's Theorem [19]: two states are probabilistically bisimilar iff they satisfy the same formulae of the logic whose formulae are  $F ::= \top \mid F \wedge F \mid [\ell]_{>r} F$ , where  $[\ell]_{>r} F$  is true at those states  $x$  where the probability  $\theta_\ell(x)(\llbracket F \rrbracket_\theta)$  of going to some state satisfying  $F$  by doing action  $\ell$  is greater than  $r$ . This is extended to any measurable space through *event bisimulations* in [5].

Mixing probabilistic (player  $P$ ) and non-deterministic ( $C$ ) behavior has also received some attention. This is notably at the heart of the *probabilistic I/O automata* of Segala and Lynch [25]. The latter can be seen as labeled Markov processes with discrete probability distributions  $\theta_\ell(x)$  (i.e., linear combinations of Dirac masses), where the set  $L$  of actions is partitioned into internal (hidden) actions and external actions. While  $P$  controls the latter, the former represent non-deterministic transitions, i.e., under the control of  $C$ . Our model of stochastic games is closer to the strictly alternating variant of probabilistic automata, where at each state, a non-deterministic choice is made among several distributions, then the next state is drawn at random according to the chosen distribution. I.e.,  $C$  plays, then  $P$ , and there is no intermediate state where  $C$  would have played

but not P. This is similar to the model by Mislove *et al.* [21], who consider state spaces that are continuous cpos. In our model, this is the other way around: in each state, P draws at random a possible choice set for C, who then picks non-deterministically from it. Additionally, our model accommodates state spaces that are discrete, or continuous cpos, or topological spaces such as  $\mathbb{R}^n$ , without any change to be made. Mislove *et al.* [21] consider a model where non-determinism is chaotic, i.e., based on a variant of Plotkin's powerdomain. We concentrate on demonic non-determinism, which is based on the Smyth powerdomain instead. For angelic non-determinism, see [14, chapitre 6], and [14, chapitre 7] for chaotic non-determinism.

Bisimulations have been studied in the above models. There are many variants on probabilistic automata [26,16,23]. Mislove *et al.* [21] show that (bi)simulation in their model is characterized by a logic similar to [8], with an added disjunction operator. Our result is similar, for a smaller logic, with one less modality. Segala and Turrini [27] compare various notions of bisimulations in these contexts.

We have already mentioned cooperative games and belief functions. See the abundant literature [6,7,28,13,24,2]. We view belief functions as generalized probabilities; the competing view as a basis for a theory of evidence is incompatible [15].

An obvious approach to studying probabilistic phenomena is to turn to measure theory and measurable spaces, see e.g. [3]. However, we hope to demonstrate that the theory of cooperative games in the case of infinite state spaces  $X$  is considerably more comfortable when  $X$  is a topological space, and we only measure opens instead of Borel subsets. This is in line with the theory of continuous valuations [17], which has had considerable success in semantics.

We use Choquet integration to integrate along capacities  $\nu$  [4]. This is exactly the notion that Tix [30] used more recently, too, and coincides with the Jones integral [17] for integration along continuous valuations. Finally, we should also note that V. Danos and M. Escardo have also come up (private comm.) with a notion of integration that generalizes Choquet integration, at least when integrating with respect to a convex game.

## 2 Preliminaries

Our state spaces  $X$  are topological spaces. We assume the reader to be familiar with (point-set) topology, in particular topology of  $T_0$  but not necessarily Hausdorff spaces. See [12,1,20] for background. Let  $\text{int}(A)$  denote the interior of  $A$ ,  $\text{cl}(A)$  its closure.

The *Scott topology* on a poset  $X$ , with ordering  $\leq$ , has as opens the upward-closed subsets  $U$  (i.e.,  $x \in U$  and  $x \leq y$  imply  $y \in U$ ) such that for every directed family  $(x_i)_{i \in I}$  having a least upper bound  $\sup_{i \in I} x_i$  inside  $U$ , some  $x_i$  is already in  $U$ . The *way-below* relation  $\ll$  is defined by  $x \ll y$  iff for any directed family  $(z_i)_{i \in I}$  with a least upper bound  $z$  such that  $y \leq z$ , then  $x \leq z_i$  for some  $i \in I$ . A poset is *continuous* iff  $\downarrow y = \{x \in X \mid x \ll y\}$  is directed, and has  $x$  as least upper bound. Then every open  $U$  can be written  $\bigcup_{x \in U} \uparrow x$ , where  $\uparrow x = \{y \in X \mid x \ll y\}$ .

Every topological space  $X$  has a specialization quasi-ordering  $\leq$ , defined by:  $x \leq y$  iff every open that contains  $x$  contains  $y$ .  $X$  is  $T_0$  iff  $\leq$  is a (partial) ordering. That of the Scott topology of a quasi-ordering  $\leq$  is  $\leq$  itself. A subset  $A \subseteq X$  is *saturated* iff  $A$  is the intersection of all opens that contain it; alternatively, iff  $A$  is upward-closed in  $\leq$ .

Every open is upward-closed. Let  $\uparrow A$  denote the upward-closure of  $A$  under a quasi-ordering  $\leq$ ,  $\downarrow A$  its downward-closure. A  $T_0$  space is *sober* iff every irreducible closed subset is the closure  $cl\{x\} = \downarrow x$  of a (unique) point  $x$ . The Hofmann-Mislove Theorem implies that every sober space is *well-filtered* [18], i.e., given any filtered family of saturated compacts  $(Q_i)_{i \in I}$  in  $X$ , and any open  $U$ ,  $\bigcap_{i \in I} Q_i \subseteq U$  iff  $Q_i \subseteq U$  for some  $i \in I$ . In particular,  $\bigcap_{i \in I} Q_i$  is saturated compact.  $X$  is *locally compact* iff whenever  $x \in U$  ( $U$  open) there is a saturated compact  $Q$  such that  $x \in int(Q) \subseteq Q \subseteq U$ . Every continuous cpo is sober and locally compact in its Scott topology. We shall consider the space  $\mathbb{R}$  of all reals with the Scott topology of its natural ordering  $\leq$ . Its opens are  $\emptyset$ ,  $\mathbb{R}$ , and the intervals  $(t, +\infty)$ ,  $t \in \mathbb{R}$ .  $\mathbb{R}$  is a stably locally compact, continuous cpo. Since we equip  $\mathbb{R}$  with the Scott topology, our *continuous functions*  $f : X \rightarrow \mathbb{R}$  are those usually called *lower semi-continuous* in the mathematical literature.

We call *capacity* on  $X$  any function  $\nu$  from  $\mathcal{O}(X)$ , the set of all opens of  $X$ , to  $\mathbb{R}^+$ , such that  $\nu(\emptyset) = 0$  (a.k.a., a *set function*.) A *game*  $\nu$  is a *monotonic capacity*, i.e.,  $U \subseteq V$  implies  $\nu(U) \leq \nu(V)$ . (The name “game” is unfortunate, as there is no obvious relationship between this and games as they are usually defined in computer science, in particular with stochastic games. The name stems from cooperative games in economics, where  $X$  is the set of players, not states.) A *valuation* is a *modular game*  $\nu$ , i.e., one such that  $\nu(U \cup V) + \nu(U \cap V) = \nu(U) \vee \nu(V)$  for every opens  $U, V$ . A game is *continuous* iff  $\nu(\bigcup_{i \in I} U_i) = \sup_{i \in I} \nu(U_i)$  for every directed family  $(U_i)_{i \in I}$  of opens. Continuous valuations have a convenient theory that fits topology well [17][18].

The *Dirac valuation*  $\delta_x$  at  $x \in X$  is the continuous valuation mapping each open  $U$  to 1 if  $x \in U$ , to 0 otherwise. (Note that  $\delta_x = u_{\{x\}}$ , by the way.) A finite linear combination  $\sum_{i=1}^n a_i \delta_{x_i}$ ,  $a_i \in \mathbb{R}^+$ , is a *simple valuation*. All simple valuations are continuous. Conversely, Jones’ Theorem [17, Theorem 5.2] states that, if  $X$  is a continuous cpo, then every continuous valuation  $\nu$  is the least upper bound  $\sup_{i \in I} \nu_i$  of a directed family  $(\nu_i)_{i \in I}$  of simple valuations way-below  $\nu$ . Continuous valuations are canonically ordered by  $\nu \leq \nu'$  iff  $\nu(U) \leq \nu'(U)$  for every open  $U$  of  $X$ .

### 3 Continuous Games, and Belief Functions

Defining the “preprobabilities” alluded to in the introduction is best done by strengthening super-additivity. A game  $\nu$  on  $X$  is *convex* iff  $\nu(U \cup V) + \nu(U \cap V) \geq \nu(U) + \nu(V)$  for every opens  $U, V$ . It is *concave* if the opposite inequality holds. Convex games are a cornerstone of economic theory. E.g., Shapley’s Theorem states that (on a finite space) the core  $\{p \text{ valuation on } X \mid \nu \leq p, \nu(X) = p(X)\}$  of any convex game  $\nu$  is non-empty, which implies the existence of economic equilibria [13][22]. But this has only been studied on discrete spaces (finiteness is implicit in [13], notably). Finite, and more generally discrete spaces are sets  $X$ , equipped with the discrete topology, so one may see our topological approach as a generalization of previous approaches.

Recall that the unanimity game  $u_A$  is defined by  $u_A(U) = 1$  if  $A \subseteq U$ ,  $u_A(U) = 0$  otherwise. Clearly,  $u_A$  is convex. It is in fact more. Call a game  $\nu$  *totally convex* (the standard name, when  $X$  is discrete, i.e., when  $U_i$  is an arbitrary subset of  $X$ , is “totally monotonic”); we changed the name so as to name total concavity the dual of total monotonicity) iff:



$$\nu \left( \bigcup_{i=1}^n U_i \right) \geq \sum_{I \subseteq \{1, \dots, n\}, I \neq \emptyset} (-1)^{|I|+1} \nu \left( \bigcap_{i \in I} U_i \right) \tag{1}$$

for every finite family  $(U_i)_{i=1}^n$  of opens ( $n \geq 1$ ), where  $|I|$  denotes the cardinality of  $I$ . A *belief function* is a totally convex game. The dual notion of *total concavity* is obtained by replacing  $\cup$  by  $\cap$  and conversely in (1), and turning  $\geq$  into  $\leq$ . A *plausibility* is a totally concave game. If  $\geq$  is replaced by  $=$  in (1), then we retrieve the familiar inclusion-exclusion principle from statistics. In particular any (continuous) valuation is a (continuous) belief function. Clearly, any belief function is a convex game. The converses of both statements fail: On  $X = \{1, 2, 3\}$  with the discrete topology,  $u_{\{1,2\}}$  is a belief function but not a valuation, and  $\frac{1}{2}(u_{\{1,2\}} + u_{\{1,3\}} + u_{\{2,3\}} - u_{\{1,2,3\}})$  is a convex game but not a belief function.

When  $X$  is finite, it is well-known [13] that any capacity  $\nu$  can be written  $\sum_{A \neq \emptyset, A \subseteq X} \alpha_A u_A$  for some coefficients  $\alpha_A \in \mathbb{R}$ , in a unique way. Also,  $\nu$  is a belief function iff all coefficients are non-negative. An interpretation of this formula is that  $\nu$  is essentially a probabilistic choice of some non-empty subset  $A$ , with probability  $\alpha_A$ , from which  $C$  can choose an element  $y \in A$  non-deterministically.

Our first result is to show that this result transfers, in some form, to the general topological case. Let  $\mathcal{Q}(X)$  be the *Smyth powerdomain* of  $X$ , i.e., the space of all non-empty compact saturated subsets  $Q$  of  $X$ , ordered by reverse inclusion  $\supseteq$ .  $\mathcal{Q}(X)$  is equipped with its Scott topology, and is known to provide an adequate model of demonic non-determinism in semantics [11]. When  $X$  is well-filtered and locally compact,  $\mathcal{Q}(X)$  is a continuous cpo. Its Scott topology is generated by the basic open sets  $\square U = \{Q \in \mathcal{Q}(X) \mid Q \subseteq U\}$ ,  $U$  open in  $X$ .

The relevance of  $\mathcal{Q}(X)$  here can be obtained by realizing that a finite linear combination  $\sum_{i=1}^n a_i u_{A_i}$  with positive coefficients is a *continuous* belief function iff every subset  $A_i$  is compact; and that  $u_{A_i} = u_{\uparrow A_i}$ . Any such linear combination that is continuous is therefore of the form  $\sum_{i=1}^n a_i u_{Q_i}$ , with  $Q_i \in \mathcal{Q}(X)$ . We call such belief functions *simple*. Returning to the interpretation above, this can be intuitively seen as a probabilistic choice of some set  $Q_i$  with probability  $a_i$ , from which  $C$  will choose  $y \in Q_i$ ; additionally,  $Q_i$  is an element of  $\mathcal{Q}(X)$ , the traditional domain for *demonic* non-determinism.

So any simple belief function  $\nu$  can be matched with a (simple) valuation  $\nu^* = \sum_{i=1}^n a_i \delta_{Q_i}$  on  $\mathcal{Q}(X)$ . Note that  $\nu^*(\square U) = \nu(U)$  for every open  $U$  of  $X$ . This is exactly the sense in which continuous belief functions are essentially continuous valuations on the space  $\mathcal{Q}(X)$  of non-deterministic choices.

**Theorem 1.** *For any continuous valuation  $P$  on  $\mathcal{Q}(X)$ , the capacity  $\nu$  defined by  $\nu(U) = P(\square U)$  is a continuous belief function on  $X$ .*

*Conversely, let  $X$  be a well-filtered and locally compact space. For every continuous belief function  $\nu$  on  $X$  there is a unique continuous valuation  $\nu^*$  on  $\mathcal{Q}(X)$  such that  $\nu(U) = \nu^*(\square U)$  for every open  $U$  of  $X$ .*

*Proof.* (Sketch.) The first part follows by computation. For the second part, observe that  $\bigcup_{i=1}^n \square U_i \subseteq \bigcup_{j=1}^m \square V_j$  iff for every  $i$ ,  $1 \leq i \leq n$ , there exists  $j$ ,  $1 \leq j \leq m$ , such that

$U_i \subseteq V_j$ . Thus, the function  $P$  given by  $P(\bigcup_{i=1}^n \square U_i) = \sum_{I \subseteq \{1, \dots, n\}, I \neq \emptyset} (-1)^{|I|+1} \nu(\bigcap_{i \in I} U_i)$  is well-defined and monotonic. Let  $\nu^*(\mathcal{U})$  be the least upper bound of  $P(\bigcup_{Q \in \mathcal{J}} \square \text{int}(Q))$ , when  $\mathcal{J}$  ranges over finite subsets of  $\mathcal{U}$ :  $\nu^*$  is monotonic, continuous,  $\nu^*(\square U) = P(\square U) = \nu(U)$ , and fairly heavy computation shows that  $\nu^*$  is modular. Uniqueness is easy.  $\square$

Next, we show that this bijection is actually an isomorphism, i.e., it also preserves order and therefore the Scott topology. To this end, define the ordering  $\leq$  on all capacities, not just valuations, by  $\nu \leq \nu'$  iff  $\nu(U) \leq \nu'(U)$  for every open  $U$  of  $X$ . We start by characterizing it in the manner of Jones' splitting lemma. This [17] Theorem 4.10] states that  $\sum_{i=1}^m a_i \delta_{x_i} \leq \sum_{j=1}^n b_j \delta_{y_j}$  iff there is matrix  $(t_{ij})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}$  of coefficients in  $\mathbb{R}^+$  such that  $\sum_{j=1}^n t_{ij} = a_i$  for each  $i$ ,  $\sum_{i=1}^m t_{ij} \leq b_j$  for each  $j$ , and whenever  $t_{ij} \neq 0$  then  $x_i \leq y_j$ . (Jones proves it for cpos, but it holds on any topological space [29] Theorem 2.4, Corollary 2.6.) We show:

**Lemma 1 (Splitting Lemma).**  $\sum_{i=1}^m a_i u_{Q_i} \leq \sum_{j=1}^n b_j u_{Q'_j}$  iff there is matrix  $(t_{ij})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}$  of coefficients in  $\mathbb{R}^+$  such that  $\sum_{j=1}^n t_{ij} = a_i$  for each  $i$ ,  $\sum_{i=1}^m t_{ij} \leq b_j$  for each  $j$ , and whenever  $t_{ij} \neq 0$  then  $Q_i \supseteq Q'_j$ .

It follows that: (A) for any two simple belief functions  $\nu, \nu'$  on  $X$ ,  $\nu \leq \nu'$  iff  $\nu^* \leq \nu'^*$ , since the two are equivalent to the existence of a matrix  $(t_{ij})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}$  satisfying the same conditions. This can be extended to all continuous belief functions, see below. Let  $\mathbf{Cd}_{\leq 1}(X)$  be the space of continuous belief functions  $\nu$  on  $X$  with  $\nu(X) \leq 1$ , ordered by  $\leq$ . Let  $\mathbf{V}_{\leq 1}(X)$  the subspace of continuous valuations. We have:

**Theorem 2.** *Let  $X$  be well-filtered and locally compact. Every continuous belief function  $\nu$  on  $X$  is the least upper bound of a directed family of simple belief functions  $\nu_i$  way-below  $\nu$ .  $\mathbf{Cd}_{\leq 1}(X)$  is a continuous cpo.*

It follows that continuous belief functions are really the same thing as (sub-)probabilities over the set of demonic choice sets  $Q \in \mathcal{Q}(X)$ .

**Theorem 3.** *Let  $X$  be well-filtered and locally compact. The function  $\nu \mapsto \nu^*$  defines an order-isomorphism from  $\mathbf{Cd}_{\leq 1}(X)$  to  $\mathbf{V}_{\leq 1}(\mathcal{Q}(X))$ .*

As a side note, (up to the  $\leq 1$  subscript)  $\mathbf{V}_{\leq 1}(\mathcal{Q}(X))$  is exactly the space into which Edalat [10] embeds a space of measures on  $\bar{X}$ . The above Theorem states that the space of objects for which we can do this is exactly  $\mathbf{Cd}_{\leq 1}(X)$ .

Dually, we may mix probabilistic choice with angelic non-determinism. Space does not permit us to describe this in detail, see [14] chapitre 6]. The point is that the space  $\mathbf{Pb}_{\leq 1}(X)$  of continuous plausibilities is order-isomorphic to  $\mathbf{V}_{\leq 1}(\mathcal{H}_u(X))$ , whenever  $X$  is stably locally compact, where the (topological) *Hoare powerdomain*  $\mathcal{H}_u(X)$  of  $X$  is the set of non-empty closed subsets of  $X$ , with the upper topology of the inclusion ordering, generated by the subbasic sets  $\diamond U = \{F \in \mathcal{H}(X) \mid F \cap U \neq \emptyset\}$ ,  $U$  open in  $X$ . The argument goes through a nice notion of convex-concave duality, which intuitively exchanges good (concave) and evil (convex). The case of chaotic non-determinism is more complex, see [14] chapitre 7].

### 4 Choquet Integration

We introduce the standard notion of integration along games  $\nu$ . This is mostly well-known [13]; adapting to the topological case is easy, so we omit proofs [14, chapitre 4].

Let  $\nu$  be a game on  $X$ , and  $f$  be continuous from  $X$  to  $\mathbb{R}$ . Recall that we equip  $\mathbb{R}$  with its Scott topology, so that  $f$  is really what is known otherwise as *lower semi-continuous*. Assume  $f$  bounded, too, i.e.,  $\inf_{x \in X} f(x) > -\infty, \sup_{x \in X} f(x) < +\infty$ . The *Choquet integral* of  $f$  along  $\nu$  is:

$$\int_{x \in X} f(x) d\nu = \int_0^{+\infty} \nu(f^{-1}(t, +\infty)) dt + \int_{-\infty}^0 [\nu(f^{-1}(t, +\infty)) - \nu(X)] dt \quad (2)$$

where both integrals on the right are improper Riemann integrals. This is well-defined, since  $f^{-1}(t, +\infty)$  is open for every  $t \in \mathbb{R}$  by assumption, and  $\nu$  measures opens. Also, since  $f$  is bounded, the improper integrals above really are ordinary Riemann integrals over some closed intervals. The function  $t \mapsto \nu(f^{-1}(t, +\infty))$  is decreasing, and every decreasing (even non-continuous, in the usual sense) function is Riemann-integrable, therefore the definition makes sense.

An alternate definition consists in observing that any *step function*  $\sum_{i=0}^n a_i \chi_{U_i}$ , where  $a_0 \in \mathbb{R}, a_1, \dots, a_n \in \mathbb{R}^+, X = U_0 \supseteq U_1 \supseteq \dots \supseteq U_n$  is a decreasing sequence of opens, and  $\chi_U$  is the indicator function of  $U$  ( $\chi_U(x) = 1$  if  $x \in U, \chi_U(x) = 0$  otherwise) is continuous, and of integral along  $\nu$  equal to  $\sum_{i=0}^n a_i \nu(U_i)$ —for *any* game  $\nu$ . It is well-known that every bounded continuous function  $f$  can be written as the least upper bound of a sequence of step functions  $f_K = a + \frac{1}{2^K} \sum_{k=1}^{\lfloor (b-a)2^K \rfloor} \chi_{f^{-1}(a + \frac{k}{2^K}, +\infty)}(x)$ ,  $K \in \mathbb{N}$ , where  $a = \inf_{x \in X} f(x), b = \sup_{x \in X} f(x)$ . Then the integral of  $f$  along  $\nu$  is the least upper bound of the increasing sequence of the integrals of  $f_K$  along  $\nu$ .

The main properties of Choquet integration are as follows. First, the integral is increasing in its function argument: if  $f \leq g$  then the integral of  $f$  along  $\nu$  is less than or equal to that of  $g$  along  $\nu$ . If  $\nu$  is continuous, then integration is also Scott-continuous in its function argument. The integral is also monotonic and Scott-continuous in the game  $\nu$ , provided the function we integrate takes only non-negative values, or provided  $\nu$  is *normalized*, i.e.,  $\nu(X) = 1$ . Integration is linear in the game, too, so integrating along  $\sum_{i=1}^n a_i \nu_i$  is the same as taking the integrals along each  $\nu_i$ , and computing the obvious linear combination. However, Choquet integration is *not* linear in the function integrated, unless the game  $\nu$  is a valuation. Still, it is *positively homogeneous*: integrating  $\alpha f$  for  $\alpha \in \mathbb{R}^+$  yields  $\alpha$  times the integral of  $f$ . It is additive on *comonotonic* functions  $f, g : X \rightarrow \mathbb{R}$  (i.e., there is no pair  $x, x' \in X$  such that  $f(x) < f(x')$  and  $g(x) > g(x')$ ). It is super-additive (the integral of  $f + g$  is at least that of  $f$  plus that of  $g$ ) when  $\nu$  is convex, in particular when  $\nu$  is a belief function, and sub-additive when  $\nu$  is concave. See [13] for the finite case, [14, chapitre 4] for the topological case.

One of the most interesting things is that integrating with respect to a unanimity game consists in taking minima. This suggests that unanimity games indeed model some *demonic* form of non-determinism. Imagine  $f(x)$  is the amount of money you gain by going to state  $x$ . The following says that taking the average amount of money with respect to a demonic adversary C will give you back the least amount possible.

**Proposition 1.** For any continuous  $f : X \rightarrow \mathbb{R}^+$ ,

$$\int_{x \in X} f(x) d\mu_A = \inf_{x \in A} f(x)$$

Moreover, if  $A$  is compact, then the inf is attained: this equals  $\min_{x \in A} f(x)$ .

Since Choquet integration is linear in the game, the integral of  $f$  along a simple belief function  $\sum_{i=1}^n a_i \mu_{Q_i}$  yields  $\sum_{i=1}^n a_i \min_{x \in Q_i} f(x)$ : this is the expected min-value of  $f$  obtained by drawing  $Q_i$  at random with probability  $a_i$  (P plays) then letting C non-deterministically move to the state  $x \in Q_i$  that minimizes the gain. We can generalize this to non-discrete probabilities over  $\mathcal{Q}(X)$  by using the  $\nu \mapsto \nu^*$  isomorphism:

**Theorem 4.** For any bounded continuous function  $f : X \rightarrow \mathbb{R}$ , let  $f_*$  be the function from  $\mathcal{Q}(X)$  to  $\mathbb{R}$  defined by  $f_*(Q) = \min_{x \in Q} f(x)$ . Say that a capacity  $\nu$  is linearly extensible from below if and only if there is continuous valuation  $P$  on  $\mathcal{Q}(X)$  with:

$$\int_{x \in X} f(x) d\nu = \int_{Q \in \mathcal{Q}(X)} f_*(Q) dP \tag{3}$$

for every bounded continuous  $f$ . If  $X$  is well-filtered and locally compact, then the capacities that are linearly extensible from below are exactly the continuous belief functions, and  $P$  must be  $\nu^*$  in [3].

It follows in particular that whenever  $\nu$  is the least upper bound of a directed family  $(\nu_i)_{i \in I}$  of simple belief functions  $\nu_i$ , then integrating  $f : X \rightarrow \mathbb{R}$  with respect to  $\nu$  can be computed by taking least upper bounds of linear combinations of mins. Therefore the Choquet integral along continuous belief functions coincides with Edalat’s lower  $R$ -integral [10], which was only defined for measures.

This can be dualized to the case of plausibilities  $\nu$ , assuming  $X$  stably locally compact [14, théorème 6.3.17]. Then we talk about capacities that are linearly extensible from above. There is an isomorphism  $\nu \mapsto \nu_*$  such that  $\nu_*(\diamond U) = \nu(U)$  for all  $U$ , and integrating  $f$  along  $\nu$  amounts to integrating  $f^*$  along  $\nu_*$ , where for every  $F \in \mathcal{H}_u(X)$ ,  $f^*(F) = \sup_{x \in F} f(x)$ . (I.e., C now maximizes our gain.) Then the Choquet integral along continuous plausibilities coincides with Edalat’s upper  $R$ -integral [10].

## 5 Ludic Transition Systems, Logic, Simulation, Rewards

Let  $\mathbf{J}_{\leq 1}(X)$  be the space of all continuous games  $\nu$  on  $X$  with  $\nu(X) \leq 1$ . This is equipped with its Scott topology. It will be practical to consider another topology. The weak topology on a subspace  $Y$  of  $\mathbf{J}_{\leq 1}(X)$  is the topology generated by the subbasic open sets  $[U > r] = \{\nu \in Y \mid \nu(U) > r\}$ ,  $U$  open in  $X$ ,  $r \in \mathbb{R}$ . It is in general coarser than the Scott topology, and coincides with it when  $Y = \mathbf{V}_{\leq 1}(X)$  and  $X$  is a continuous cpo [30, Satz 4.10]. One can show that the weak topology is exactly the coarsest that makes continuous all functionals mapping  $\nu \in Y$  to the integral of  $f$  along  $\nu$ , for all  $f : X \rightarrow \mathbb{R}^+$  bounded continuous. (See [14, section 4.5] for details.)

By analogy with Markov kernels and LMPs, define a ludic transition system as a family  $\theta = (\theta_\ell)_{\ell \in L}$ , where  $L$  is a given set of actions, and each  $\theta_\ell$  is a continuous map from the state space  $X$  to  $\mathbf{J}_{\leq 1} \text{ wk}(X)$ . (See [14, chapitres 8, 9] for missing details.) The

main change is that, as announced in the introduction, we replace probability distributions by continuous games. One may object that LMPs are defined as *measurable*, not continuous, so that this definition overly restricts the class of transition systems we are considering. However, the mathematics are considerably cleaner when assuming continuity. Moreover, the weak topology is so weak that, for example, it only restrains  $\theta_\ell$  so that  $x \mapsto \theta_\ell(x)(U)$  is continuous as a function from  $X$  to  $\mathbb{R}^+$ , equipped with its *Scott* topology; this certainly allows it to have jumps. Finally, one may argue, following Edalat [10], that any second countable locally compact Hausdorff space  $X$  can be embedded as a set of maximal elements of a continuous cpo (namely  $\mathcal{Q}(X)$ ; other choices are possible) so that any measure on  $X$  extends to a continuous valuation on  $\mathcal{Q}(X)$ . This provides a theory of approximation of integration on  $X$  through domain theory. One may hope a similar phenomenon will apply to games—for some notion of games yet to be defined on Borel subsets, not opens.

**Logic.** Following [85], define the logic  $\mathcal{L}_{\text{open}}^{\top \wedge \vee}$  by the grammar shown right, where  $\ell \in L, r \in \mathbb{Q} \cap [0, 1]$  in the last line. Compared to [85], we only have one extra disjunction operator. The same logic, with disjunction, is shown to characterize simulation for LMPs in [9, Section 2.3].

|                 |                   |
|-----------------|-------------------|
| $F ::= \top$    | true              |
| $F \wedge F$    | conjunction (and) |
| $F \vee F$      | disjunction (or)  |
| $[\ell]_{>r} F$ | modality          |

Let  $\llbracket F \rrbracket_\theta$  be the set of states  $x \in X$  where  $F$  holds:  $\llbracket \top \rrbracket_\theta = X, \llbracket F_1 \wedge F_2 \rrbracket_\theta = \llbracket F_1 \rrbracket_\theta \wedge \llbracket F_2 \rrbracket_\theta, \llbracket F_1 \vee F_2 \rrbracket_\theta = \llbracket F_1 \rrbracket_\theta \vee \llbracket F_2 \rrbracket_\theta,$  and  $\llbracket [\ell]_{>r} F \rrbracket_\theta = \delta_\ell^{-1}[\llbracket F \rrbracket_\theta > r]$  is the set of states  $x$  such that the preprobability  $\delta_\ell(\llbracket F \rrbracket_\theta)$  that the next state  $y$  will satisfy  $F$  on firing an  $\ell$  action is strictly greater than  $r$ . Note that this is well-defined, precisely because  $\delta_\ell$  is continuous from  $X$  to a space of games with the weak topology. Also, it is easy to see that  $\llbracket F \rrbracket_\theta$  is always open.

**Simulation.** Now define *simulation* in the spirit of event bisimulation [5] (we shall see below why we do not call it *bisimulation*). For any topology  $\mathcal{O}$  on  $X$  coarser than that of  $X$ , let  $X : \mathcal{O}$  be  $X$  equipped with the topology  $\mathcal{O}$ . A *simulation topology* for  $\theta$  is a topology  $\mathcal{O}$  on  $X$ , coarser than that of  $X$ , such that  $\delta_\ell$  is continuous from  $X : \mathcal{O}$  to  $\mathbf{J}_{\leq 1}^{wk}(X : \mathcal{O})$ , i.e.,  $\delta_\ell^{-1}[U > r] \in \mathcal{O}$  for each  $U \in \mathcal{O}$  and each  $r \in \mathbb{R}$ . (A close notion was introduced in [31, Theorem 29].) One non-explanation for this definition is to state that this is exactly event bisimulation [5], only replacing  $\sigma$ -algebras by topologies. A better explanation is to revert back to Larsen and Skou’s original definition of probabilistic bisimulation in terms of an algebra of *tests* (in slightly more abstract form). A (bi)simulation should not be thought as an arbitrary equivalence relation, rather as one generated from a collection  $Tst$  of tests, which are subsets  $A$  of  $X$ :  $x \in X$  passes the test iff  $x \in A$ , it fails it otherwise. Two elements are equivalent iff they pass the same tests. Now in a continuous setting it only makes sense that the tests be open: any open  $U$  defines a continuous predicate  $\chi_U$  from  $X$  to the Sierpiński space  $\mathbb{S} = \{0, 1\}$  (with the Scott topology of  $0 \leq 1$ ), and conversely. Let  $\mathcal{O}_{Tst}$  be the topology generated by the tests  $Tst$ . It is sensible to require that  $\delta_\ell^{-1}[U > r]$  be a test, too, at least when  $U$  is a finite union of finite intersections of tests (for the general case, appeal to the fact that  $\delta_\ell(x)$  is continuous, and that any open can be approximated by such a finite union): one can indeed test whether  $x \in \delta_\ell^{-1}[U > r]$  by firing transitions according to the

preprobability  $\delta_\ell(x)$ , and test (e.g., by sampling, knowing that if  $\delta_\ell(x)$  is a belief function for example, then we are actually playing also against a demonic adversary  $\mathbb{C}$ ) whether our chances of getting to a state  $y \in U$  exceed  $r$ . And this is essentially how we defined simulation topologies.

Every simulation topology  $\mathcal{O}$  defines a specialization quasi-ordering  $\preceq_{\mathcal{O}}$ , which is the analogue of the standard notion of simulation here. (Note that in the case of event bisimulation, i.e., taking  $\sigma$ -algebras instead of topologies,  $\preceq_{\mathcal{O}}$  would be an equivalence relation—because  $\sigma$ -algebras are closed under complements—justifying the fact that event bisimulation really is a bisimulation, while our notion is a simulation.) Write  $\equiv_{\mathcal{O}} = \preceq_{\mathcal{O}} \cap \succeq_{\mathcal{O}}$  the equivalence associated with simulation  $\preceq_{\mathcal{O}}$ . Clearly, there is a coarsest (largest) simulation topology  $\mathfrak{D}_{\theta}$ . The following is then easy:

**Theorem 5.** *Let  $\mathcal{O}$  be a simulation topology for  $\theta$  on  $X$ . For any  $F \in \mathcal{L}_{open}$ ,  $\llbracket F \rrbracket_{\theta} \in \mathcal{O}$ . In particular [Soundness], if  $x \in \llbracket F \rrbracket_{\theta}$  and  $x \preceq_{\mathcal{O}} y$  then  $y \in \llbracket F \rrbracket_{\theta}$ . Conversely [Completeness], the coarsest simulation topology  $\mathfrak{D}_{\theta}$  is exactly that generated by the opens  $\llbracket F \rrbracket_{\theta}$ ,  $F \in \mathcal{L}_{open}^{\top \wedge \vee}$ .*

This can be used, as is standard in the theory of Markov chains, to lump states. Given a topology  $\mathcal{O}$ , let  $X/\mathcal{O}$  be the quotient space  $X/\equiv_{\mathcal{O}}$ , equipped with the finest topology such that  $q_{\mathcal{O}} : X \rightarrow X/\mathcal{O}$  is continuous. Let the direct image  $f[\nu]$  of a game  $\nu$  on  $X$  by a continuous map  $f : X \rightarrow Y$  be  $f[\nu](V) = \nu(f^{-1}(V))$ . Taking direct images preserves monotonicity, modularity, (total) convexity, (total) concavity, and continuity.

**Proposition 2.** *Let  $\mathcal{O}$  be a simulation topology for  $\theta$ . The function  $\theta_{\ell}/\mathcal{O}$  mapping  $q_{\mathcal{O}}(x)$  to  $q_{\mathcal{O}}[\theta_{\ell}(x)]$  is well defined and continuous from  $X/\mathcal{O}$  to  $\mathbf{J}_{\leq 1}^{wk}(X/\mathcal{O})$  for every  $\ell \in L$ . The family  $\theta/\mathcal{O} = (\theta_{\ell}/\mathcal{O})_{\ell \in L}$  is then a ludic transition system on  $X/\mathcal{O}$ , which we call the lumped ludic transition system.*

For any  $F \in \mathcal{L}_{open}^{\top \wedge \vee}$  and  $x \in X$ ,  $x$  and  $q_{\mathcal{O}}(x)$  satisfy the same formulae:  $q_{\mathcal{O}}(\llbracket F \rrbracket_{\theta}) = \llbracket F \rrbracket_{\theta/\mathcal{O}}$ , and  $\llbracket F \rrbracket_{\theta} = q_{\mathcal{O}}^{-1}(\llbracket F \rrbracket_{\theta/\mathcal{O}})$ , in particular,  $x \in \llbracket F \rrbracket_{\theta}$  iff  $q_{\mathcal{O}}(x) \in \llbracket F \rrbracket_{\theta/\mathcal{O}}$ .

**Rewards and payoffs.** A classical problem on Markov decision processes is to evaluate average payoffs. Since LMPs and ludic transition systems are so similar, we can do exactly the same. Imagine  $P$  plays according to a finite-state program  $\Pi$ , i.e., an automaton with internal states  $q, q'$  and transitions  $q \xrightarrow{\ell} q'$ . Let  $r_{q \xrightarrow{\ell} q'} : X \rightarrow \mathbb{R}$  be a family of bounded continuous reward functions: we may think that  $r_{q \xrightarrow{\ell} q'}(x)$  is the amount of money  $P$  gains if she fires her internal transition  $q \xrightarrow{\ell} q'$ , drawing the next state  $y$  at random along  $\theta_{\ell}(x)$ . Let  $\gamma_{q \xrightarrow{\ell} q'} \in (0, 1]$  be a family of so-called discounts. Define the average payoff, starting from state  $x$  when  $P$  is in its internal state  $q$ , by:

$$V_q(x) = \sup_{\ell, q'/q \xrightarrow{\ell} q'} \left[ r_{q \xrightarrow{\ell} q'}(x) + \gamma_{q \xrightarrow{\ell} q'} \int_{y \in X} V_{q'}(y) d\theta_{\ell}(x) \right] \tag{4}$$

This formula would be standard if  $\theta_\ell(x)$  were a probability distribution. What is less standard is what (4) means when  $\theta_\ell(x)$  is a game. E.g., when  $\theta_\ell(x)$  is a simple belief function  $\sum_{i=1}^{n_\ell} a_{i\ell x} u_{Q_{i\ell x}}$ , then:

$$V_q(x) = \sup_{\ell, q' / q \xrightarrow{\ell} q'} \left[ r_{q \xrightarrow{\ell} q'}(x) + \gamma_{q \xrightarrow{\ell} q'} \sum_{i=1}^{n_\ell} a_{i\ell x} \min_{y \in Q_{i\ell x}} V_{q'}(y) \right] \tag{5}$$

where we see that P has control over the visible transitions  $\ell$ , and tries to maximize his payoff (sup), while C will minimize it, and some averaging is taking place in-between. The equation (4) does not always have a solution in the family of all  $V_q$ s. But there are two cases where it has, similar to those encountered in Markov decision processes.

**Theorem 6.** *Assume  $\theta$  is standard, i.e.,  $\theta_\ell(X)$  is always either 0 or 1, and the set  $\{x \in X \mid \theta_\ell(x) = 0\}$  of deadlock states is open; or that  $r_{q \xrightarrow{\ell} q'}(x) \geq 0$  for all  $q, \ell, q', x \in X$ . Assume also that there are  $a, b \in \mathbb{R}$  with  $a \leq r_{q \xrightarrow{\ell} q'}(x), \gamma_{q \xrightarrow{\ell} q'} \leq b$  for all  $q, \ell, q', x \in X$ . Then (4) has a unique solution in any of the following two cases:*

*[Finite Horizon] If all paths in  $\Pi$  have bounded length.*

*[Discount] If there is a constant  $\gamma \in (0, 1)$  such that  $\gamma_{q \xrightarrow{\ell} q'} \leq \gamma$  for every  $q, \ell, q'$ .*

When  $\theta_\ell$  is a simple belief function, Equation (5) is then a Bellman-type equation that can be solved by dynamic programming techniques. Then observe that any continuous belief function is the directed lub of simple belief functions by Theorem 2 under mild assumptions. This offers a canonical way to approximate the average payoff  $V_q$ .

**Acknowledgments.** Thanks to F. Laviolette, J. Desharnais, V. Danos, P. Panangaden, Ph. Scott, M. Escardo, and the many others who expressed their support. Thanks to the anonymous referees for their helpful comments.

## References

1. Abramsky, S., Jung, A.: Domain theory. In: Abramsky, S., Gabbay, D.M., Maibaum, T.S.E. (eds.) Handbook of Logic in Computer Science, vol. 3, pp. 1–168. Oxford University Press, Oxford (1994)
2. Bassett Jr., G.W., Koenker, R., Kordas, G.: Pessimistic portfolio allocation and Choquet expected utility (January 2004) Available from <http://www.econ.uiuc.edu/~roger/research/risk/choquet.pdf>
3. Cattani, S., Segala, R., Kwiatkowska, M.Z., Norman, G.: Stochastic transition systems for continuous state spaces and non-determinism. In: Sassone, V. (ed.) FOSSACS 2005. LNCS, vol. 3441, pp. 125–139. Springer, Heidelberg (2005)
4. Choquet, G.: Theory of capacities. Annales de l’Institut Fourier 5, 131–295 (1953–54)
5. Danos, V., Desharnais, J., Laviolette, F., Panangaden, P.: Bisimulation and cocongruence for probabilistic systems. Information and Computation 204(4), 503–523 (2006) Special issue for selected papers from CMCS04, 22 pages.
6. Dempster, A.P.: Upper and lower probabilities induced by a multivalued mapping. Annals of Mathematical Statistics 38, 325–339 (1967)
7. Dempster, A.P.: A generalization of Bayesian inference. Journal of the Royal Statistical Society B 30, 205–247 (1968)

8. Desharnais, J., Edalat, A., Panangaden, P.: Bisimulation for labelled Markov processes. *Information and Computation* 179(2), 163–193 (2002)
9. Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Approximating labeled Markov processes. *Information and Computation* 184(1), 160–200 (2003)
10. Edalat, A.: Domain theory and integration. *Theoretical Computer Science* 151, 163–193 (1995)
11. Feinberg, E.A., Schwartz, A.: *Handbook of Markov Decision Processes, Methods and Applications*, pages. 565. Kluwer Academic Publishers, Dordrecht (2002)
12. Gierz, G., Hofmann, K.H., Keimel, K., Lawson, J.D., Mislove, M., Scott, D.S.: *A Compendium of Continuous Lattices*. Springer, Heidelberg (1980)
13. Gilboa, I., Schmeidler, D.: Additive representation of non-additive measures and the Choquet integral. Discussion Papers 985. Center for Mathematical Studies in Economics and Management Science, Northwestern University (1992)
14. Goubault-Larrecq, J.: Une introduction aux capacités, aux jeux et aux prévisions, 516 pages (January 2007)  
<http://www.lsv.ens-cachan.fr/~goubault/ProNobis/pp.pdf>
15. Halpern, J.Y., Fagin, R.: Two views of belief: Belief as generalized probability and belief as evidence. *Artificial Intelligence* 54, 275–317 (1992)
16. Hansson, H.A., Jonsson, B.: A calculus for communicating systems with time and probabilities. In: Hans, A. (ed.) *Proc. 11th IEEE Real-time Systems Symp.*, Silver Spring, MD, pp. 278–287. IEEE Computer Society Press, Los Alamitos (1990)
17. Jones, C.: Probabilistic Non-Determinism. PhD thesis, University of Edinburgh, Technical Report ECS-LFCS-90-105 (1990)
18. Jung, A.: Stably compact spaces and the probabilistic powerspace construction. In: Desharnais, J., Panangaden, P. (eds.) *Domain-theoretic Methods in Probabilistic Processes*. *Electronic Lecture Notes in Computer Science*, vol. 87, pp. 15. Elsevier, Amsterdam (2004)
19. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. *Information and Computation* 94, 1–28 (1991)
20. Mislove, M.: Topology, domain theory and theoretical computer science. *Topology and Its Applications* 89, 3–59 (1998)
21. Mislove, M., Ouaknine, J., Worrell, J.: Axioms for probability and nondeterminism. In: *Proc. 10th Int. Workshop on Expressiveness in Concurrency (EXPRESS'03)*. *Electronic Notes in Theoretical Computer Science*, vol. 91(3), pp. 7–28 (2003)
22. Osborne, M.J., Rubinstein, A.: *A Course in Game Theory*. MIT Press, Cambridge (1994)
23. Philippou, A., Lee, I., Sokolsky, O.: Weak bisimulation for probabilistic processes. In: Palamidessi, C. (ed.) *CONCUR 2000*. LNCS, vol. 1877, pp. 334–349. Springer, Heidelberg (2000)
24. Schmeidler, D.: Subjective probability and expected utility without additivity. *Econometrica* 57, 571–587 (1989)
25. Segala, R.: *Modeling and Verification of Randomized Distributed Real-Time Systems*. MIT Press, Cambridge, MA (1996)
26. Segala, R., Lynch, N.: Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing* 2(2), 250–273 (1995)
27. Segala, R., Turrini, A.: Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic models. In: *2nd Int. Conf. Quantitative Evaluation of Systems (QEST 2005)*, Torino, Italy, September 2005, pp. 44–53. IEEE Computer Society Press, Los Alamitos (2005)
28. Shafer, G.: *A Mathematical Theory of Evidence*. Princeton University Press, Princeton (1976)



29. Philipp, S.: Spaces of valuations as quasimetric domain. In: Edalat, A., Jung, A., Keimel, K., Kwiatkowska, M. (eds.) Proceedings of the 3rd Workshop on Computation and Approximation (Comprox III), Birmingham, England, September 1997. Electronic Notes in Theoretical Computer Science, vol. 13, Elsevier, Amsterdam (1997)
30. Tix, R.: Stetige Bewertungen auf topologischen Räumen. Diplomarbeit, TH Darmstadt (June 1995)
31. van Breugel, F., Mislove, M., Ouaknine, J., Worrell, J.: Domain theory, testing and simulation for labelled markov processes. Theoretical Computer Science 333(1-2), 171–197 (2005)

# On the Chromatic Number of Random Graphs

Amin Coja-Oghlan<sup>1,\*</sup>, Konstantinos Panagiotou<sup>2,\*\*</sup>, and Angelika Steger<sup>2</sup>

<sup>1</sup> Department of Mathematical Sciences, Carnegie Mellon University  
Pittsburgh PA15213, USA

amincoja@andrew.cmu.edu

<sup>2</sup> Institute of Theoretical Computer Science ETH Zentrum, Universitätsstr. 6  
CH - 8092 Zurich, Switzerland  
{panagiok, steger}@inf.ethz.ch

**Abstract.** In this paper we study the chromatic number  $\chi(G_{n,p})$  of the binomial random graph  $G_{n,p}$ , where  $p = p(n) \leq n^{-3/4-\delta}$ , for every fixed  $\delta > 0$ . We prove that a.a.s.  $\chi(G_{n,p})$  is  $\ell$ ,  $\ell + 1$ , or  $\ell + 2$ , where  $\ell$  is the maximum integer satisfying  $2(\ell - 1) \log(\ell - 1) \leq np$ .

## 1 Introduction and Results

*Coloring random graphs.* Let  $G = (V, E)$  be a graph. A  $k$ -coloring of  $G$  is an assignment  $V \rightarrow \{1, \dots, k\}$  of colors to the vertices, such that adjacent vertices receive distinct colors. Moreover, the *chromatic number*  $\chi(G)$  of  $G$  is the least integer  $k$  such that  $G$  admits a  $k$ -coloring.

Determining the chromatic number of a given graph is a fundamental though notoriously hard computational problem. Indeed, Feige and Kilian [1] proved that no polynomial time algorithm approximates  $\chi(G)$  within a factor of  $|V|^{1-\epsilon}$ , unless  $\mathcal{NP} = \mathcal{ZPP}$  ( $\epsilon > 0$  arbitrarily small but independent of  $G$ ). Hence, there are no efficient algorithms that perform well on *every* graph coloring instance (unless  $\mathcal{NP} = \mathcal{ZPP}$ ). Therefore, a large number of authors have studied graph coloring *heuristics*, i.e., efficient algorithms that are supposed to yield “reasonable” results on “most” instances (where “reasonable” and “most” are not always well-defined in a mathematical sense), see e.g. [2] or [3] and references therein.

In order to compare graph coloring heuristics, it is important to consider meaningful benchmark instances. Among the test instances that are in common use are various types of *random graphs*. In fact, the Erdős-Rényi model  $G_{n,p}$ , which is one of the most studied random graph models, provides very interesting and – for certain values of the parameters – extremely “difficult” benchmark instances. The random graph  $G_{n,p}$  is defined as follows: the vertex set is  $V = \{1, \dots, n\}$ , and each of the  $\binom{n}{2}$  possible edges among two vertices in  $V$  is present with probability  $0 < p = p(n) < 1$  independently. Thus, the expected number of edges in  $G_{n,p}$  is  $\binom{n}{2}p$ , and the expected degree of a vertex is  $(n - 1)p$ . We will say that  $G_{n,p}$  has some property  $\mathcal{A}$  *asymptotically almost surely* (“a.a.s.”), if the probability that  $G_{n,p}$  has  $\mathcal{A}$  tends to 1 as  $n \rightarrow \infty$ .

---

\* Supported by the German Research Foundation (grant CO 646).

\*\* Supported by the SNF, grant number: 200021-107880/1.

The random graph  $G_{n,p}$  and several variants of this model have been used to provide rigorous analyses of graph coloring heuristics. Krivelevich [4] provides a thorough survey of these results. For more current results we refer the reader to Achlioptas and Moore [5], where an extension of a list-coloring algorithm is analysed, and to Jia and Moore [6], where a backtracking algorithm is investigated in great detail, and references therein.

The empirically best current heuristic for coloring  $G_{n,p}$  is *Survey Propagation* [7,8]. This algorithm and its analysis are based on mathematically non-rigorous arguments from statistical physics. Moreover, these methods also yield exciting conjectures about the *probable value(s) of  $\chi(G_{n,p})$*  and concerning the “solution space geometry”, i.e., the number and relative position of optimal colorings of  $G_{n,p}$ .

*The chromatic number of  $G_{n,p}$ .* Concerning the probable value of  $\chi(G_{n,p})$ , the considerations in [7] suggest that if  $p = \frac{c}{n}$ , where  $c$  is sufficiently large, then a.a.s.  $\chi(G_{n,p}) \leq \ell + 1$ , where

$$\ell = \ell(n, p) = \max\{l \in \mathbb{N} : 2(l - 1) \log(l - 1) \leq np\}. \tag{1}$$

In a remarkable paper, Achlioptas and Naor [9] proved rigorously this statement. In fact, for edge probabilities  $\frac{c}{n}$ , they proved for roughly half of all  $c \in (0, \infty)$  that a.a.s. “ $\chi(G_{n,p}) = \ell + 1$ ”, and the slightly weaker statement “ $\chi(G_{n,p}) \in \{\ell, \ell + 1\}$ ” for the remaining  $c$ ’s. Thus, the result of Achlioptas and Naor deals with random graphs of *bounded* average degree, and yields the probable value of  $\chi(G_{n,p})$  up to an additive error of only *one* in this case.

The main result of the present paper is the following theorem, which deals with random graphs of *unbounded* average degree.

**Theorem 1.** *Let  $0 < \delta \leq \frac{1}{4}$ ,  $\frac{1}{n} \leq p = p(n) \leq n^{-3/4-\delta}$ , and let  $\ell = \ell(n, p)$  be as in [1]. Then  $\chi(G_{n,p}) \in \{\ell, \ell + 1, \ell + 2\}$  a.a.s. Furthermore, for every fixed  $\varepsilon > 0$ , if  $np \in ((2\ell - 1) \log \ell + \varepsilon, 2\ell \log \ell)$ , then  $\chi(G_{n,p}) \in \{\ell + 1, \ell + 2\}$  a.a.s.*

Hence, Theorem 1 yields the probable value of  $\chi(G_{n,p})$  up to an additive error of at most *two* for random graphs of average degree up to  $n^{1/4-\delta}$ , and is a natural extension to the main theorem of [9].

*Related work.* Since the seminal work of Erdős and Rényi [10], computing the probable value of  $\chi(G_{n,p})$  has been a fundamental problem in the theory of random graphs. Bollobás [11] was the first to obtain an asymptotically tight result: he showed that if  $0 < p < 1$  is fixed, then

$$\chi(G_{n,p}) \sim -\frac{n \log(1 - p)}{2 \log(np)} \quad \text{a.a.s.} \tag{2}$$

Moreover, Łuczak [12] extended (2) to the regime  $\frac{1}{n} \ll p = o(1)$ , proving that

$$\left| \chi(G_{n,p}) - \frac{np}{2 \log(np)} \right| = O\left(\frac{np \cdot \log \log np}{\log^2 np}\right) \quad \text{a.a.s.} \tag{3}$$

Thus, while (3) shows that  $\chi(G_{n,p}) \sim \frac{np}{2 \log(np)}$ , the additive error term  $\mathcal{O}(\frac{np \log \log np}{\log^2 np})$  is unbounded if the average degree  $np \rightarrow \infty$ . Hence, Theorem 1 provides a significantly tighter estimate than (3), which is the best previous result.

In addition to its probable value, also the concentration of  $\chi(G_{n,p})$  has received considerable attention. Shamir and Spencer [13] proved that  $\chi(G_{n,p})$  is concentrated on  $\mathcal{O}(\sqrt{n})$  integers for any sequence  $p = p(n)$  of edge probabilities. Furthermore, they showed that  $\chi(G_{n,p})$  is concentrated in an interval of constant length for  $p \ll n^{-1/2}$ . Moreover, Łuczak [12] proved that  $\chi(G_{n,p})$  is concentrated on two consecutive integers if  $p \ll n^{-5/6}$ . Finally, Alon and Krivelevich [14] proved that two point concentration actually holds under the weaker assumption  $p \ll n^{-1/2}$ , which is best possible in the sense that there are  $p = p(n)$  for which  $\chi(G_{n,p})$  is not concentrated on one value. However, none of these papers [14][12][13] yields the specific values on which  $\chi(G_{n,p})$  is concentrated. For instance, while Alon and Krivelevich show that for each  $p = p(n)$  there exists a sequence  $r = r(n, p)$  such that a.a.s. it holds  $\chi(G_{n,p}) \in \{r, r + 1\}$ , the proof does not yield any clue on what the value of  $r$  is.

*Techniques and outline.* The proof of Theorem 1 builds on and extends some of the techniques from [9][14][12][13]. Suppose that  $n^{-1} \ll p \leq n^{-3/4-\delta}$ , and let  $\ell$  be as in (1). To bound  $\chi(G_{n,p})$  from below, we just verify that the expected number of  $(\ell - 1)$ -colorings is  $o(1)$ , so that Markov’s inequality yields that  $\chi(G_{n,p}) \geq \ell$  a.a.s.

Following Achlioptas and Naor [9], we employ the *second moment method* to bound  $\Pr[\chi(G_{n,p}) \leq \ell + 1]$  from below. That is, we estimate the second moment  $\mathbb{E}[X^2]$  of the number  $X$  of  $(\ell + 1)$ -colorings of  $G_{n,p}$ ; this estimate employs a general result from [9] on optimizing certain functions over stochastic matrices (Theorem 2 below). Since  $\Pr[X > 0] \geq \frac{\mathbb{E}[X]^2}{\mathbb{E}[X^2]}$ , the upper bound on  $\mathbb{E}[X^2]$  gives us a lower bound for the probability that  $\chi(G_{n,p}) \leq \ell + 1$ . More precisely, in Section 3 we shall prove that

$$\Pr[\chi(G_{n,p}) \leq \ell + 1] \geq e^{-6(pn)^2} \cdot n^{-\ell^2}. \tag{4}$$

Now, the obvious problem is that the r.h.s. of the “lower bound” (4) actually tends to 0 as  $n \rightarrow \infty$ . This problem does not occur in the *sparse* regime considered in [9] (where  $np = c$  is constant). Indeed, in the sparse regime it is true that  $\Pr[\chi(G_{n,p}) \leq \ell + 1] \geq \alpha(c)$ , where  $\alpha(c)$  remains bounded away from 0 as  $n \rightarrow \infty$  (of course, this does not follow from (4)). Therefore, Achlioptas and Naor can boost this lower bound using a sharp threshold result of Achlioptas and Friedgut [15], thus concluding that actually  $\chi(G_{n,p}) \leq \ell + 1$  a.a.s.

However, in the case that  $np \rightarrow \infty$ , which is the main focus of the present work, we cannot bound  $\Pr[\chi(G_{n,p}) \leq \ell + 1]$  away from 0 uniformly as  $n \rightarrow \infty$ . In addition, the sharp threshold result [15] does not apply. Nevertheless, adapting arguments from Shamir and Spencer [13], in Section 4 we shall prove that a.a.s.  $G = G_{n,p}$  admits a set  $U$  of vertices of size  $|U| \leq n^{3/2} p \log n$  such that  $\chi(G \setminus U) \leq \ell + 1$ . Thus, to prove that  $\chi(G_{n,p}) \leq \ell + 2$  a.a.s. we just need to show

that any such partial  $(\ell + 1)$ -coloring can be modified, such that by spending one additional color, we can construct a  $(\ell + 2)$ -coloring of the entire graph a.a.s.

To this end, we consider two cases. If  $np \leq n^{1/20}$ , say, then a slight variation of Łuczak’s argument [12] yields that  $\chi(G) \leq \ell + 2$  a.a.s. By contrast, the case  $n^{1/20} \leq np \ll n^{1/4}$  requires new ideas: extending tools developed by Alon and Krivelevich [14], we show the following. Suppose that an  $(\ell + 1)$ -coloring of  $G \setminus U$  is given. Then one can recolor a few vertices in  $G \setminus U$  with one new color so that the resulting  $(\ell + 2)$ -coloring of  $G \setminus U$  can be extended to an  $(\ell + 2)$ -coloring of all of  $G$ . Thus, to color the vertices in  $U$  we recolor a few vertices of  $G \setminus U$  and reuse some of the “old” colors to color  $U$ .

The above argument relates to the proof of Alon and Krivelevich as follows. In [14], it is assumed that there is a set  $W \subset V$  of size  $|W| \leq \sqrt{n} \log n$  such that  $\chi(G \setminus W) \leq k$  for a certain number  $k$ , and the goal is to prove that then  $\chi(G) \leq k + 1$ . By contrast, in the present paper the “exceptional” set  $U$  has size  $n^{3/2} p \log n \gg \sqrt{n} \log n$ . Thus, we need to extend the coloring to a significantly larger number of “exceptional” vertices and study the combinatorial structure of  $G$  more precisely.

Some proofs are omitted due to space constraints – they can be found in the full version of the paper [16].

## 2 Preliminaries and Notation

Let  $G = (V, E)$  be a graph, and  $X, Y \subseteq V$ . We shall denote by  $e(X, Y)$  the number of edges in  $G$  with one endpoint in  $X$  and one endpoint in  $Y$ . Furthermore, for every  $v \in V$  we will denote by  $\Gamma(v)$  the neighbors of  $v$  in  $G$ , and by  $\Gamma(X) := \bigcup_{v \in X} \Gamma(v)$ .

Before we prove our main result (Theorem 1) in the following sections, we shall introduce a few technical tools, which will be used extensively in the sequel. These tools can be found for instance in [17] or [18]. The first lemma is a well-known estimate for the tail of the binomial distribution.

**Lemma 1 (Chernoff bounds).** *Let  $X$  be a binomially distributed variable with  $\lambda := \mathbb{E}[X]$ . For every  $t \geq 0$  it holds*

$$\Pr[X \geq \lambda + t] \leq e^{-\frac{t^2}{2(\lambda+t/3)}}, \quad \text{and} \quad \Pr[X \leq \lambda - t] \leq e^{-\frac{t^2}{2\lambda}}.$$

The next lemma is a special case of a far more general result based on martingale inequalities. We present this version here, as it suffices for our intended application.

**Lemma 2 (Azuma/Hoeffding’s inequality).** *Let  $f$  be a function on graphs, such that  $|f(G) - f(G')| \leq 1$ , whenever  $G$  and  $G'$  differ only in edges adjacent to the same vertex. Then the random variable  $X = f(G_{n,p})$  with  $\lambda := \mathbb{E}[X]$  satisfies*

$$\Pr[X \geq \lambda + t] \leq e^{-\frac{2t^2}{n}}, \quad \text{and} \quad \Pr[X \leq \lambda - t] \leq e^{-\frac{2t^2}{n}}.$$

Finally, we are going to exploit the following version of the Lovasz Local Lemma, which provides us with a lower bound for the probability of the non-occurrence of certain events.

**Lemma 3 (Lovász Local Lemma, symmetric version).** *Let  $A_1, \dots, A_n$  be events in an arbitrary probability space. Suppose that each event  $A_i$  is mutually independent of a set of all other events  $A_j$  but of at most  $d$ , and that  $\Pr[A_i] \leq p$  for all  $1 \leq i \leq n$ . If  $ep(d + 1) \leq 1$ , then  $\Pr[\bigwedge_{i=1}^n \overline{A_i}] > 0$ .*

### 3 Approaching the Values of the Chromatic Number

Let  $G_{n,m}$  be a random graph on  $n$  labeled vertices and  $m$  edges, drawn uniformly at random from the set of all such graphs. In this section we shall derive a lower bound for  $\chi(G_{n,m})$ , which holds with high probability, and an upper bound for  $\chi(G_{n,p})$ , which holds with some probability, that we can bound from below.

**Proposition 1.** *Let  $n \in \mathbb{N}$  and  $\ell = \ell(n)$  be an integer function of  $n$  such that  $\ell \leq \frac{n^{1/2}}{\log n}$ . For every fixed  $\varepsilon > 0$ , if  $d \geq \frac{\log \ell}{\log \ell - \log(\ell - 1)} + \varepsilon$ ,  $G_{n,dn}$  is a.a.s. not  $\ell$ -colorable.*

The proof is based on estimating the expectation of the number  $X$  of colorings of  $G_{n,m}$  with  $\ell$  colors, and by using the inequality  $\Pr[X > 0] \leq \mathbb{E}[X]$ . Observe that for  $\ell \rightarrow \infty$ , it holds  $\frac{\log \ell}{\log \ell - \log(\ell - 1)} = (\ell - \frac{1}{2}) \log \ell + o(1)$ . Before we proceed with the proof of the upper bound for  $\chi(G_{n,p})$ , let us introduce a tool which plays a crucial role in our arguments. Let  $\mathcal{S}_\ell$  denote the set of  $\ell \times \ell$  row-stochastic matrices, i.e., matrices from  $[0, 1]^{\ell \times \ell}$ , such that the values of the rows sum up to one. For  $M \in \mathcal{S}_\ell$ , let

$$\mathcal{H}(M) := -\frac{1}{\ell} \sum_{1 \leq i, j \leq \ell} m_{ij} \log m_{ij} \text{ and } \mathcal{E}(M) := \log \left( 1 - \frac{2}{\ell} + \frac{1}{\ell^2} \sum_{1 \leq i, j \leq \ell} m_{ij}^2 \right), \tag{5}$$

and define the function  $g_d(M) := \mathcal{H}(M) + d\mathcal{E}(M)$ . In our proof we will exploit the following general result by Achlioptas and Naor [9].

**Theorem 2.** *Let  $\ell \in \mathbb{N}$  and  $J_\ell$  be the constant  $\ell \times \ell$  matrix, whose entries are all equal to  $\frac{1}{\ell}$ . If  $d \leq (\ell - 1) \log(\ell - 1)$ , then for all  $M \in \mathcal{S}_\ell$  it holds  $g_d(J_\ell) \geq g_d(M)$ .*

**Proposition 2.** *Let  $n \in \mathbb{N}$  and  $\ell$  be an integer function of  $n$  such that  $\ell \leq \frac{n^{1/2}}{\log n}$ . Let  $C_{\ell,m}$  denote the random variable, which counts the number of colorings of  $G_{n,m}$  with  $\ell$  colors. If  $\frac{m}{n} \leq (\ell - 1) \log(\ell - 1)$ , then  $\Pr[C_{\ell,m} > 0] \geq e^{-17(\frac{m}{n})^2} \cdot n^{-\ell^2}$ .*

From the above proposition we obtain easily the following lemma for the binomial random graph, as the models  $G_{n,p}$  and  $G_{n,m}$  behave similarly when  $m \approx p\binom{n}{2}$ .

**Lemma 4.** *Let  $0 < \delta \leq \frac{1}{2}$  and  $p = p(n) \leq n^{-1/2-\delta}$ . Then the following statement is true for sufficiently large  $n$ . Let  $C_{\ell,p}$  be the number of colorings of  $G_{n,p}$*

with  $\ell$  colors. If  $\ell$  is the maximum integer satisfying  $2(\ell - 2) \log(\ell - 2) \leq p(n - 1)$ , then

$$\Pr[C_{\ell,p} > 0] \geq e^{-6(pn)^2} \cdot n^{-2\ell^2}. \tag{6}$$

*Proof (Proof of Proposition 2).* In order to prove the statement, we use similar ideas as in [9]. An important difference here is that  $\ell$  is a function of  $n$  (instead of being constant), and our contribution is that we take into account how it modifies the involved constants (which now become functions of  $\ell$ ) in the original proof. Furthermore, we are working directly with the uniform random graph  $G_{n,m}$  which does not have any multiple edges or loops.

Let  $d := \frac{m}{n}$  and denote by  $B_\ell$  the number of “balanced” colorings of  $G_{n,dn}$ , where balanced means that the sizes of all color classes are either  $\lfloor \frac{n}{\ell} \rfloor$  or  $\lceil \frac{n}{\ell} \rceil$ . For the sake of exposition, we shall omit in the remainder floors and ceilings. As the number of edges not connecting vertices in the same color class is  $\frac{\ell-1}{2\ell}n^2$ , by using  $1 - x \geq e^{-2x}$ , valid for small  $x$ , the probability that a balanced partition is a valid coloring is for sufficiently large  $n$

$$\frac{\left(\frac{\ell-1}{2\ell}n^2\right)^{\frac{dn}{2}}}{\binom{n}{2}^{\frac{dn}{2}}} \geq \frac{\left(\frac{\ell-1}{2\ell}n^2 - dn\right)^{\frac{dn}{2}}}{\left(\frac{n^2}{2}\right)^{\frac{dn}{2}}} \geq \left(1 - \frac{1}{\ell}\right)^{dn} e^{-\frac{4\ell}{\ell-1}d^2} \geq \left(1 - \frac{1}{\ell}\right)^{dn} \cdot e^{-8d^2}.$$

By applying Stirling’s formula  $1 \leq n! / \left(\frac{n}{e}\right)^n \sqrt{2\pi n} \leq 2$  we obtain easily

$$\mathbb{E}[B_\ell] \geq \frac{n!}{\left(\frac{n}{\ell}\right)!^\ell} \cdot \left(1 - \frac{1}{\ell}\right)^{dn} \cdot e^{-8d^2} \geq (2\pi n)^{-\frac{\ell-1}{2}} \cdot \ell^{\frac{\ell}{2}} \cdot \left[\ell \cdot \left(1 - \frac{1}{\ell}\right)^d\right]^{2n} \cdot e^{-8d^2}.$$

In the following we are going to argue that

$$\mathbb{E}[B_\ell^2] \leq e^{6d} \cdot n^{\ell^2 - \ell} \cdot \left[\ell \cdot \left(1 - \frac{1}{\ell}\right)^d\right]^{2n}, \tag{7}$$

which will complete the proof for large  $n$ , as  $\Pr[B_\ell > 0] \geq \mathbb{E}[B_\ell]^2 / \mathbb{E}[B_\ell^2]$ . In order to calculate  $\mathbb{E}[B_\ell^2]$ , it is sufficient to consider *pairs* of balanced partitions, and to bound the probability that both are simultaneously valid colorings. Let  $\Pi = (V_1, \dots, V_\ell)$  and  $\Pi' = (V'_1, \dots, V'_\ell)$  be two partitions, and define  $d_{ij}(\Pi, \Pi') := |V_i \cap V'_j|$ . The probability that an edge is bichromatic in both  $\Pi$  and  $\Pi'$  is proportional to the number of edges, that do not join vertices in one of the color classes  $V_1, \dots, V_\ell$  or  $V'_1, \dots, V'_\ell$ . The number of such edges is precisely  $\binom{n}{2} - 2\ell \binom{n/\ell}{2} + \sum \binom{d_{ij}}{2}(\Pi, \Pi')$ . Hence, the probability that both  $\Pi$  and  $\Pi'$  are valid colorings is for sufficiently large  $n$

$$\begin{aligned} \frac{\left(\binom{n}{2} - 2\ell \binom{n/\ell}{2} + \sum \binom{d_{ij}}{2}\right)^{\frac{dn}{2}}}{\binom{n}{2}^{\frac{dn}{2}}} &\leq \frac{\left(\binom{n}{2} - 2\ell \binom{n/\ell}{2} + \sum \binom{d_{ij}}{2}\right)^{dn}}{\binom{n}{2}^{dn}} \\ &\leq e^{6d} \left(1 - \frac{2}{\ell} + \sum \left(\frac{d_{ij}}{n}\right)^2\right)^{dn}. \end{aligned}$$

Abbreviate  $1 - \frac{2}{\ell} + \sum(\frac{d_{ij}}{n})^2 := q$ , and let  $\mathcal{D}$  be the set of matrices with non-negative integer entries, where all rows and columns sum up to  $\frac{n}{\ell}$ . By using  $(\frac{x}{e})^x \leq x! \leq 10\sqrt{x}(\frac{x}{e})^x$ , we obtain

$$\mathbb{E} [B_\ell^2] \leq e^{6d} \cdot \sum_{D \in \mathcal{D}} \frac{n!}{\prod_{1 \leq i, j \leq \ell} d_{ij}!} q^{dn} \leq e^{6d} \cdot \sqrt{n} \cdot \sum_{D \in \mathcal{D}} e^{n(\mathcal{H}(\frac{\ell}{n}D) + \log \ell + d\mathcal{E}(\frac{\ell}{n}D))},$$

where  $\mathcal{H}$  and  $\mathcal{E}$  are defined in (5). Due to Theorem 2, we have for all  $D \in \mathcal{D}$

$$\mathcal{H}\left(\frac{\ell}{n}D\right) + d\mathcal{E}\left(\frac{\ell}{n}D\right) \leq \mathcal{H}(J_\ell) + d\mathcal{E}(J_\ell) = \log \ell + d \log \left(1 - \frac{2}{\ell} + \frac{1}{\ell^2}\right).$$

As the number of matrices in  $\mathcal{D}$  is at most  $n^{(\ell-1)^2}$ , we obtain

$$\mathbb{E} [B_\ell^2] \leq e^{6d} \cdot \sqrt{n} \cdot n^{(\ell-1)^2} \cdot e^{n(2 \log \ell + d \log((1 - \frac{1}{\ell})^2))} = e^{6d} \cdot n^{\ell^2 - \ell} \cdot \ell^{2n} \cdot \left(1 - \frac{1}{\ell}\right)^{2dn}.$$

□

## 4 Proof of the Main Result

### 4.1 The Sparse Case ( $n^{-1} \ll p \ll n^{-1 + \frac{1}{20}}$ )

Our first lemma follows directly from [12], Fact 2.

**Lemma 5.** *Let  $n^{-1} \ll p \ll n^{-1 + \frac{1}{20}}$ . Every subset  $U$  of the vertex set of  $G_{n,p}$ , such that  $|U| \leq n^{3/4}$ , spans less than  $(\frac{3}{2} - \frac{1}{9})|U|$  edges.*

The next lemma states essentially that a.a.s.  $G_{n,p}$  is almost  $(\ell + 1)$ -colorable, where  $\ell$  is given by (1).

**Lemma 6.** *Let  $n^{-1} \leq p \leq n^{-3/4}$  and let  $\ell$  be the maximum integer satisfying  $2(\ell - 2) \log(\ell - 2) \leq p(n - 1)$ . Then a.a.s. there is a subset  $U_0 = U_0(G_{n,p})$  of the vertex set of  $G_{n,p}$  of size at most  $n^{3/2}p \log n$ , such that  $G_{n,p} \setminus U_0$  is  $\ell$ -colorable.*

The above lemma states that if we choose  $\ell$  as prescribed, then all vertices of  $G_{n,p}$  are colorable with the colors  $\{1, \dots, \ell + 1\}$ , except for a small set  $U_0$ . In the remainder we shall assume that  $G = G_{n,p}$  satisfies the assertions of Lemma 5 and 6 and we are going to argue that by using only one additional color, we can color  $U_0$  such that we obtain a valid coloring for the whole graph.

To achieve this, we construct a set  $U \supseteq U_0$  of size at most  $n^{3/4}$ , such that  $I = \Gamma(U) \setminus U$  is stable. For such a set  $U$ , we can color  $G$  with  $\ell + 2$  colors. Indeed,  $U$  is 3-colorable, as due to Lemma 5 all its subsets have average degree less than 3. Now color  $U$  with the colors  $\{1, 2, 3\}$ , and  $I$  with a fresh color  $\ell + 2$  (c.f. Figure 1).

To obtain  $U$ , we begin with  $U_0$ , and extend it by two vertices  $x, y$  in its neighborhood, if  $\{x, y\} \in G$ . Now observe that this process stops with  $|U| \leq n^{3/4}$ , as otherwise the number of edges joining vertices in  $U$  would be for sufficiently large  $n$  greater than  $3 \frac{|U| - |U_0|}{2} \geq (\frac{3}{2} - \frac{1}{9})|U|$ , contradicting Lemma 5.



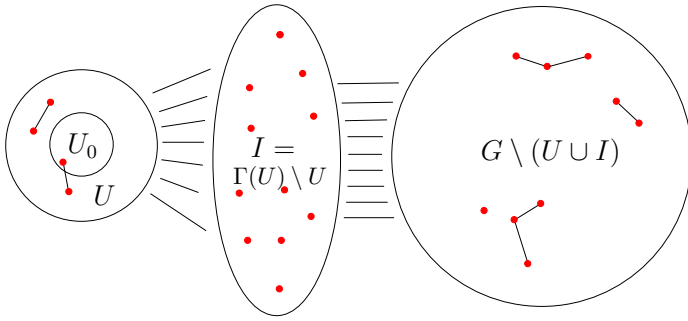


Fig. 1. Coloring  $G = G_{n,p}$  in the sparse case

4.2 The Dense Case ( $n^{-1+\frac{1}{20}} \leq p \leq n^{-\frac{3}{4}-\delta}$ )

In this section we assume that  $p = p(n) = n^{-3/4-\delta}$ , where  $0 < \delta \leq \frac{1}{4} - \frac{1}{20}$ . In order to deal with edge probabilities  $p$  in this regime, and thus with denser random graphs than in the previous section, we have to extend the above argument significantly. Let  $\ell = \ell(p, n)$  be the maximum integer satisfying  $2(\ell - 2) \log(\ell - 2) \leq p(n - 1)$ , and note that a straightforward calculation yields  $\ell \geq \frac{np}{3 \log(np)}$ .

A graph  $G$  with vertex set  $V$  is called  $d$ -choosable, if for every family  $\mathcal{C} = \{S_v \in \mathbb{N}^d \mid v \in V\}$  of sets of colors for the vertices of  $G$ , there exists a proper vertex coloring  $c : V \rightarrow \mathbb{N}$ , such that  $c(v) \in S_v$  for all  $v$ . The lemma below states that every not too large subset of vertices of the  $G_{n,p}$  spans very few edges, if  $p$  is substantially smaller than  $n^{-3/4}$ ; furthermore, every such subset is  $t$ -choosable, for a small constant  $t$ . The proof is based on standard random graph arguments – we omit the details.

**Lemma 7.** *Let  $p = p(n) = n^{-3/4-\delta}$ , where  $0 < \delta \leq \frac{1}{4} - \frac{1}{20}$ .  $G_{n,p}$  has a.a.s. the following properties. Every subset  $U$  of the vertex set of  $G_{n,p}$ , such that  $|U| \leq n^{3/4}$ , spans at most  $\lceil \frac{1}{\delta} \rceil |U|$  edges. Furthermore, every  $U$  is  $2\lceil \frac{1}{\delta} \rceil$ -choosable.*

Similarly as before, we shall see that a.a.s. the random graph  $G = G_{n,p}$  admits a set  $U \subset V$  of size  $|U| \leq 2n^{3/2}p \log n$ , such that  $\chi(G \setminus U) \leq \ell$ , and such that  $U$  is only “sparsely connected” to  $V \setminus U$ . However, we cannot guarantee that  $\Gamma(U) \setminus U$  is a stable set. Instead, we shall recolor a few vertices in  $V \setminus U$  with a new “joker color” in such a way that we can then reuse some of the colors of the  $\ell$ -coloring of  $G \setminus U$  to color the vertices in  $U$ . Hence, we will just need one additional color, so that  $\chi(G) \leq \ell + 1$  a.a.s.

We now elaborate the details of this approach. As a first step, we show that there is a subset  $U$  of the vertex set of size  $|U| \leq 2n^{3/2}p \log n$ , such that a.a.s.  $\chi(G \setminus U) \leq \ell$ , and every vertex in  $V \setminus U$  has at most a constant number of neighbors in  $U$ .

**Lemma 8.** *Let  $p = p(n) = n^{-3/4-\delta}$ , where  $0 < \delta \leq \frac{1}{4} - \frac{1}{20}$ , and let  $\ell$  be the maximum integer satisfying  $2(\ell - 2) \log(\ell - 2) \leq p(n - 1)$ .  $G_{n,p}$  has a.a.s. the*

following property. There is a set of vertices  $U = U(G_{n,p})$  of size  $< 2n^{3/2}p \log n$  such that  $\chi(G \setminus U) \leq \ell$ , and every vertex in  $V \setminus U$  has at most  $\xi := 50\lceil \frac{1}{\delta} \rceil$  neighbors in  $U$ .

*Proof.* According to Lemma 6, a.a.s. there exists a set  $U_0$  of size at most  $n^{3/2}p \log n$ , such that  $\chi(G_{n,p} \setminus U_0) \leq \ell$ . Moreover, according to Lemma 7, every subset  $X$  of the vertices of size at most  $n^{3/4}$  spans a.a.s. at most  $\lceil \frac{1}{\delta} \rceil |X|$  edges. We assume that  $G = G_{n,p}$  has these properties. To obtain  $U$ , we start with  $U_0$ , and enhance it iteratively with vertices, which violate the desired condition. This process stops with  $|U| < 2n^{3/2}p \log n$ , as otherwise we would get a subset  $U$  of the vertices of  $G$  with  $|U| = 2n^{3/2}p \log n \leq n^{3/4}$ , that spans more than  $(|U| - |U_0|)\xi \geq 50\lceil \frac{1}{\delta} \rceil n^{3/2}p \log n > \lceil \frac{1}{\delta} \rceil |U|$  edges – a contradiction.  $\square$

Let  $c : V \setminus U \rightarrow \{1, \dots, \ell\}$  be a  $\ell$ -coloring of  $V \setminus U$ . In order to obtain a  $(\ell + 1)$ -coloring  $c^* : V \rightarrow \{1, \dots, \ell + 1\}$  of the entire graph, we shall recolor some of the vertices in  $V \setminus U$  with the new color  $\ell + 1$ , so that we can reuse the old colors  $1, \dots, \ell$  to color the vertices in  $U$ . More precisely, our strategy is as follows. As every subset of vertices of size at most  $n^{3/4}$  is a.a.s.  $\eta := 2\lceil \frac{1}{\delta} \rceil$ -choosable, we shall assign to each vertex  $u \in U$  a list  $L_u \subset \{1, \dots, \ell\}$  of colors such that the following holds: let

$$\Gamma_u := \{v \in \Gamma(u) \setminus U : c(v) \in L_u\}$$

be the set of all neighbors in  $V \setminus U$  of  $u \in U$ , whose color lies in  $L_u$ . Then  $\Gamma_U := \bigcup_{u \in U} \Gamma_u$  is a *stable set*.

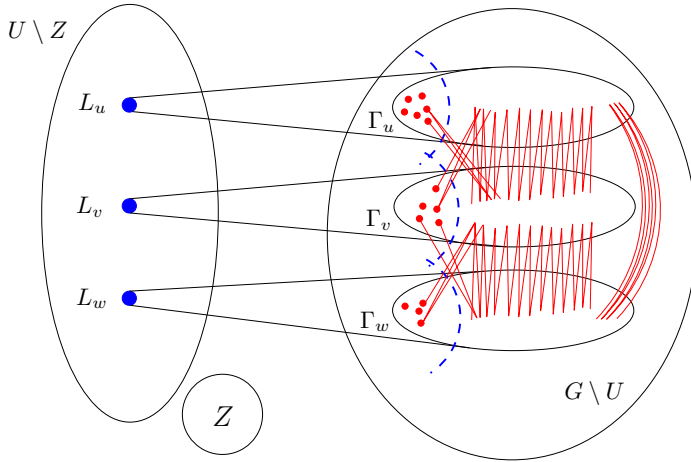
If we could exhibit such lists  $(L_u)_{u \in U}$ , then it would be easy to obtain a  $(\ell + 1)$ -coloring  $c^*$  of  $G$ : color the vertices in  $\Gamma_U$  with the “joker color”  $\ell + 1$ , thereby making the colors in  $L_u$  available for  $u \in U$ . Then, color each vertex in  $u \in U$  with a color from  $L_u$ , which is possible due to Lemma 7. For an illustration, see Figure 2.

Hence, the remaining task is to show that a.a.s. there exist lists  $(L_u)_{u \in U}$  with the desired property. In this context, Alon and Krivelevich [14] proved the following.

**Lemma 9.** *Let  $p = p(n) = n^{-3/4-\delta}$ , where  $0 < \delta \leq \frac{1}{4} - \frac{1}{20}$ .  $G_{n,p}$  has a.a.s. the following property. Assume that  $R \subset V$  is such that  $G_{n,p} \setminus R$  has a  $t$ -coloring  $c : V \setminus R \rightarrow \{1, \dots, t\}$ . Moreover, suppose that  $Q \subset R$  is a set of size  $|Q| \leq \sqrt{n}$ . Then there exist lists  $(L_u)_{u \in Q}$  of colors  $1, \dots, t$  such that  $|L_u| \geq \eta := 2\lceil \frac{1}{\delta} \rceil$  for all  $u \in Q$  and  $\Gamma_Q = \{v \in \Gamma(u) \setminus Q : c(v) \in L_u, u \in Q\}$  is stable.*

Due to the previous discussion, the above lemma implies that if  $R = Q$  (i.e. if we could find such a subset of  $G_{n,p}$  of size  $\leq \sqrt{n}$ ), then  $G_{n,p}$  would be  $(t + 1)$ -colorable. In fact, Proposition 3.1 of [14] claims just this consequence. The version that we stated above follows directly from their proof – they show that the claim of Lemma 9 holds, and proceed as just described.

Unfortunately, for our intended application, Lemma 9 is not strong enough. Lemma 8 only yields a set  $U$  of size  $\leq 2n^{3/2}p \log n$ , whereas Lemma 9 requires that  $|Q| \leq \sqrt{n} \ll 2n^{3/2}p \log n$ . Therefore, to construct the lists  $(L_u)_{u \in U}$ , we



**Fig. 2.** Coloring  $G = G_{n,p}$  in the dense case. The lists  $L_x$  are chosen such that the union of the vertices  $\Gamma_x$  in the corresponding color classes in  $G \setminus U$  form an independent set.

extend the approach of Alon and Krivelevich as follows. We shall show that up to a small “exceptional set”  $Z \subset U$  all vertices in  $v \in U \setminus Z$  have the property that their neighborhood  $\Gamma(v) \setminus U$  outside of  $U$  is only sparsely connected to the neighborhoods  $\Gamma(U) \setminus U$  of the remaining vertices in  $U$ . This will enable us to apply the Lovász Local Lemma to prove in a probabilistic fashion that such lists  $L_u$  for  $u \in U \setminus Z$  always exist. Furthermore, the exceptional vertices in  $Z$  will be considered separately: as  $|Z| \leq \sqrt{n}$ , we can just apply Lemma 9 with  $Q = Z$  and  $R = U$  to obtain the lists  $L_u$  for  $u \in Z$ .

The following lemma yields the desired small exceptional set  $Z$ .

**Lemma 10.** *Let  $p = p(n) = n^{-3/4-\delta}$ , where  $0 < \delta \leq \frac{1}{4} - \frac{1}{20}$ .  $G_{n,p}$  enjoys the following property a.a.s. For every subset  $U$  of the vertices of size  $1 \ll |U| \leq 2n^{3/2}p \log n$ , such that every  $v \notin U$  has at most  $\xi := 50 \lceil \frac{1}{\delta} \rceil$  neighbors in  $U$ , there exists  $Z \subseteq U$  of size  $|Z| \leq n^{-1/2}p^{-1}(\log n)^6$  such that  $v \in U \setminus Z$  satisfies*

$$e(\Gamma(v) \setminus U, \Gamma(U \setminus Z) \setminus U) \leq \xi^{-7} \left( \frac{np}{\log(np)} \right)^2, \tag{8}$$

$$e(\Gamma(v) \setminus U, \Gamma(Z) \setminus U) \leq \xi^{-7} \frac{np}{\log(np)}. \tag{9}$$

The proof of the above lemma can be found in the full version of the paper [16]. Applying Lemma 9 to  $Q = Z$  and  $R = U$ , we obtain lists  $(L_u)_{u \in Z} \subset \{1, \dots, \ell\}$  of colors such that  $|L_u| \geq \eta$ , and such that the set  $\Gamma_Z$  of neighbors  $v \in V \setminus U$  of vertices  $u \in Z$  whose color  $c(v)$  belongs to  $L_u$  is stable.

As a final step, we assign lists  $L_u$  to the vertices  $u \in U \setminus Z$ . For each vertex  $u \in U \setminus Z$  we consider the set

$$F_u := \{c(v) : v \in \Gamma_Z \text{ is adjacent to some } w \in \Gamma(u) \setminus U\}.$$

Hence,  $F_u$  consists of all colors in  $\{1, \dots, \ell\}$ , that we do not want to include in the list  $L_u$ , as otherwise we would generate a conflict with a vertex  $v \in \Gamma(Z) \setminus U$ , that will have the “joker color” in  $c^*$  due to the choice of the lists for the vertices in  $Z$ . Note that (9) implies

$$|F_u| \leq \frac{np}{\xi^7 \log(np)} \stackrel{(\ell \geq \frac{np}{3 \log(np)})}{\leq} \frac{\ell}{2} \quad \text{for all } u \in U \setminus Z. \tag{10}$$

Now, independently for all  $u \in U \setminus Z$  we choose a list  $L_u \subset \{1, \dots, \ell\} \setminus F_u$  of size  $|L_u| = \eta$  uniformly at random.

Letting  $\Gamma_u = \{v \in \Gamma(u) : c(v) \in L_u\}$ ,  $\Gamma_U = \bigcup_{u \in U} \Gamma_u$ , and  $\Gamma_{U \setminus Z} = \bigcup_{u \in U} \Gamma_u$ , we complete the proof by showing that with positive probability  $\Gamma_U$  is stable. The crucial ingredient is the following lemma.

**Lemma 11.** *With the above assumptions, the probability that  $\Gamma_{U \setminus Z}$  is stable (taken over the choice of the random lists  $(L_u)_{u \in U \setminus Z}$ ) is positive.*

Lemma 11 implies that there is a way to choose the lists  $(L_u)_{u \in U \setminus Z}$  such that  $\Gamma_{U \setminus Z}$  is stable, because a randomly chosen list of sets has this property with positive probability. Further, we already know that  $\Gamma_Z = \bigcup_{u \in Z} \Gamma_u$  is stable (by Lemma 9). Moreover,  $G$  contains no  $\Gamma_{U \setminus Z}$ - $\Gamma_Z$  edges, because in the construction of the lists  $(L_u)_{u \in U \setminus Z}$  we have forbidden all colors  $F_u$  that might yield such edges. Thus, we have established that  $\Gamma_U = \Gamma_{U \setminus Z} \cup \Gamma_Z$  is stable, thereby completing the proof.

## References

1. Feige, U., Kilian, J.: Zero knowledge and the chromatic number. *J. Comput. System Sci.* 57(2), 187–199 (1998)
2. Zeitlhofer, T., Wess, B.: A comparison of graph coloring heuristics for register allocation based on coalescing in interval graphs. In: *ISCAS* (4), 529–532 (2004)
3. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partitioning. *Oper. Res.* 39(3), 378–406 (1991)
4. Krivelevich, M.: Coloring random graphs—an algorithmic perspective. In: *Mathematics and computer science, II* (Versailles, 2002). *Trends Math.* Birkhäuser, Basel, pp. 175–195 (2002)
5. Achlioptas, D., Moore, C.: Almost all graphs with average degree 4 are 3-colorable. *J. Comput. System Sci.* 67(2), 441–471 (2003), Special issue on STOC’02 (Montreal, QC)
6. Jia, H., Moore, C.: How much backtracking does it take to color random graphs? Rigorous results on heavy tails (2004)

7. Mulet, R., Pagnani, A., Weigt, M., Zecchina, R.: Coloring random graphs. *Physical Review Letters* 89, 268701 (2002)
8. Braunstein, A., Mulet, R., Pagnani, A., Weigt, M., Zecchina, R.: Polynomial iterative algorithms for coloring and analyzing random graphs. *Physical Review E* 68, 36702 (2003)
9. Achlioptas, D., Naor, A.: The two possible values of the chromatic number of a random graph. *Ann. of Math. (2)* 162(3), 1335–1351 (2005)
10. Erdős, P., Rényi, A.: On random graphs. I. *Publ. Math. Debrecen* 6, 290–297 (1959)
11. Bollobás, B.: The chromatic number of random graphs. *Combinatorica* 8(1), 49–55 (1988)
12. Luczak, T.: A note on the sharp concentration of the chromatic number of random graphs. *Combinatorica* 11(3), 295–297 (1991)
13. Shamir, E., Spencer, J.: Sharp concentration of the chromatic number on random graphs  $G_{n,p}$ . *Combinatorica* 7(1), 121–129 (1987)
14. Alon, N., Krivelevich, M.: The concentration of the chromatic number of random graphs. *Combinatorica* 17(3), 303–313 (1997)
15. Achlioptas, D., Friedgut, E.: A sharp threshold for  $k$ -colorability. *Random Structures Algorithms* 14(1), 63–70 (1999)
16. <http://www.ti.inf.ethz.ch/as/people/panagiotou/papers/CNoSRG.ps>
17. Alon, N., Spencer, J.H.: The probabilistic method, 2nd edn. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience [John Wiley & Sons], Chichester (2000)
18. Janson, S., Luczak, T., Rucinski, A.: Random graphs. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, Chichester (2000)

# Quasi-randomness and Algorithmic Regularity for Graphs with General Degree Distributions

Noga Alon<sup>1,\*</sup>, Amin Coja-Oghlan<sup>2,3,\*\*</sup>, Hiệp Hàn<sup>3,\*\*\*</sup>, Mihyun Kang<sup>3,†</sup>,  
Vojtěch Rödl<sup>4,‡</sup>, and Mathias Schacht<sup>3,§</sup>

<sup>1</sup> School of Mathematics and Computer Science, Raymond and Beverly Sackler  
Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel  
noga@math.tau.ac.il

<sup>2</sup> Carnegie Mellon University, Department of Mathematical Sciences, Pittsburgh,  
PA 15213, USA

<sup>3</sup> Humboldt-Universität zu Berlin, Institut für Informatik, Unter den Linden 6, 10099  
Berlin, Germany

{coja,hhan,kang,schacht}@informatik.hu-berlin.de

<sup>4</sup> Department of Mathematics and Computer Science, Emory University, Atlanta,  
GA 30322, USA

rodl@mathcs.emory.edu

**Abstract.** We deal with two very related subjects: quasi-randomness and regular partitions. The purpose of the concept of quasi-randomness is to measure how much a given graph “resembles” a random one. Moreover, a regular partition approximates a given graph by a bounded number of quasi-random graphs. Regarding quasi-randomness, we present a new spectral characterization of low discrepancy, which extends to sparse graphs. Concerning regular partitions, we present a novel concept of regularity that takes into account the graph’s degree distribution, and show that if  $G = (V, E)$  satisfies a certain boundedness condition, then  $G$  admits a regular partition. In addition, building on the work of Alon and Naor [4], we provide an algorithm that computes a regular partition of a given (possibly sparse) graph  $G$  in polynomial time.

**Keywords:** quasi-random graphs, Laplacian eigenvalues, sparse graphs, regularity lemma, Grothendieck’s inequality.

## 1 Introduction and Results

This paper deals with quasi-randomness and regular partitions. Loosely speaking, a graph is quasi-random if the global distribution of the edges resembles

---

\* Supported by an ISF grant, and by the Hermann Minkowski Minerva Center for Geometry at Tel Aviv University.

\*\* Supported by DFG grant FOR 413/1-2 and COJ 646.

\*\*\* Supported by DFG in the research training group *Methods for Discrete Structures*.

† Supported by DFG grant PR 296/7-3.

‡ Supported by NSF Grant DMS 0300529.

§ Supported by DFG grant SCHA 1263/1-1 and GIF Grant 889/05.

the expected edge distribution of a random graph. Furthermore, a regular partition approximates a given graph by a constant number of quasi-random graphs; such partitions are of algorithmic importance, because a number of NP-hard problems can be solved in polynomial time on graphs that come with regular partitions. In this section we present our main results. References to related work can be found in Section 2, and the remaining sections contain proof sketches and detailed descriptions of the algorithms.

**Quasi-Randomness: discrepancy and eigenvalues.** Random graphs are well known to have a number of remarkable properties (e.g., excellent expansion). Therefore, quantifying how much a given graph “resembles” a random graph is an important problem, both from a structural and an algorithmic point of view. Providing such measures is the purpose of the notion of *quasi-randomness*. While this concept is rather well developed for dense graphs (i.e., graphs  $G = (V, E)$  with  $|E| = \Omega(|V|^2)$ ), less is known in the sparse case, which we deal with in the present work. In fact, we shall actually deal with (sparse) graphs with *general degree distributions*, including but not limited to the ubiquitous power-law degree distributions (cf. [1]).

We will mainly consider two types of quasi-random properties: low discrepancy and eigenvalue separation. The low discrepancy property concerns the global edge distribution and basically states that *every* set  $S$  of vertices approximately spans as many edges as we would expect in a random graph with the same degree distribution. More precisely, if  $G = (V, E)$  is a graph, then we let  $d_v$  signify the degree of  $v \in V$ . Furthermore, the *volume* of a set  $S \subset V$  is  $\text{vol}(S) = \sum_{v \in S} d_v$ . In addition,  $e(S)$  denotes the number of edges spanned by  $S$ .

Disc( $\varepsilon$ ): We say that  $G$  has *discrepancy at most  $\varepsilon$*  (“ $G$  has Disc( $\varepsilon$ )” for short) if

$$\forall S \subset V : \left| e(S) - \frac{\text{vol}(S)^2}{2\text{vol}(V)} \right| < \varepsilon \cdot \text{vol}(V). \tag{1}$$

To explain (1), let  $\mathbf{d} = (d_v)_{v \in V}$ , and let  $G(\mathbf{d})$  signify a uniformly distributed random graph with degree distribution  $\mathbf{d}$ . Then the probability  $p_{vw}$  that two vertices  $v, w \in V$  are adjacent in  $G(\mathbf{d})$  is proportional to the degrees of both  $v$  and  $w$ , and hence to their product. Further, as the total number of edges is determined by the sum of the degrees, we have  $\sum_{(v,w) \in V^2} p_{vw} = \text{vol}(V)$ , whence  $p_{vw} \sim d_v d_w / \text{vol}(V)$ . Therefore, in  $G(\mathbf{d})$  the *expected* number of edges inside of  $S \subset V$  equals  $\frac{1}{2} \sum_{(v,w) \in S^2} p_{vw} \sim \frac{1}{2} \text{vol}(S)^2 / \text{vol}(V)$ . Consequently, (1) just says that for *any* set  $S$  the actual number  $e(S)$  of edges inside of  $S$  must not deviate from what we expect in  $G(\mathbf{d})$  by more than an  $\varepsilon$ -fraction of the total volume.

An obvious problem with the bounded discrepancy property (1) is that it is quite difficult to check whether  $G = (V, E)$  satisfies this condition. This is because one would have to inspect an exponential number of subsets  $S \subset V$ . Therefore, we consider a second property that refers to the eigenvalues of a certain matrix representing  $G$ . More precisely, we will deal with the *normalized*

Laplacian  $L(G)$ , whose entries  $(\ell_{vw})_{v,w \in V}$  are defined as

$$\ell_{vw} = \begin{cases} 1 & \text{if } v = w \text{ and } d_v \geq 1, \\ -(d_v d_w)^{-\frac{1}{2}} & \text{if } v, w \text{ are adjacent,} \\ 0 & \text{otherwise;} \end{cases}$$

$L(G)$  turns out to be appropriate for representing graphs with general degree distributions.

**Eig( $\delta$ ):** Letting  $0 = \lambda_1(L(G)) \leq \dots \leq \lambda_{|V|}(L(G))$  denote the eigenvalues of  $L(G)$ , we say that  $G$  has  $\delta$ -eigenvalue separation (“ $G$  has Eig( $\delta$ )”) if  $1 - \delta \leq \lambda_2(L(G)) \leq \lambda_{|V|}(L(G)) \leq 1 + \delta$ .

As the eigenvalues of  $L(G)$  can be computed in polynomial time (within arbitrary numerical precision), we can essentially check efficiently whether  $G$  has Eig( $\delta$ ) or not.

It is not difficult to see that Eig( $\delta$ ) provides a *sufficient* condition for Disc( $\varepsilon$ ). That is, for any  $\varepsilon > 0$  there is a  $\delta > 0$  such that any graph  $G$  that has Eig( $\delta$ ) also has Disc( $\varepsilon$ ). However, while the converse implication is true if  $G$  is dense (i.e.,  $\text{vol}(V) = \Omega(|V|^2)$ ), it is false for sparse graphs. In fact, providing a *necessary* condition for Disc( $\varepsilon$ ) in terms of eigenvalues has been an open problem in the area of sparse quasi-random graphs since the work of Chung and Graham [8]. Concerning this problem, we basically observe that the reason why Disc( $\varepsilon$ ) does in general not imply Eig( $\delta$ ) is the existence of a small set of “exceptional” vertices. With this in mind we refine the definition of Eig as follows.

**ess-Eig( $\delta$ ):** We say  $G$  has *essential  $\delta$ -eigenvalue separation* (“ $G$  has ess-Eig( $\delta$ )”) if there is a set  $W \subset V$  of volume  $\text{vol}(W) \geq (1 - \delta)\text{vol}(V)$  such that the following is true. Let  $L(G)_W = (\ell_{vw})_{v,w \in W}$  denote the minor of  $L(G)$  induced on  $W \times W$ , and let  $\lambda_1(L(G)_W) \leq \dots \leq \lambda_{|W|}(L(G)_W)$  signify its eigenvalues; then we require that  $1 - \delta < \lambda_2(L(G)_W) < \lambda_{|W|}(L(G)_W) < 1 + \delta$ .

**Theorem 1.** *There is a constant  $\gamma > 0$  such that the following is true for all graphs  $G = (V, E)$  and all  $\varepsilon > 0$ .*

1. *If  $G$  has ess-Eig( $\varepsilon$ ), then  $G$  satisfies Disc( $10\sqrt{\varepsilon}$ ).*
2. *If  $G$  has Disc( $\gamma\varepsilon^3$ ), then  $G$  satisfies ess-Eig( $\varepsilon$ ).*

The proof of Theorem 1 is based on Grothendieck’s inequality and the duality theorem for semidefinite programs. In effect, the proof actually provides us with an efficient algorithm that computes a set  $W$  as in the definition of ess-Eig( $\varepsilon$ ), provided that the input graph has Disc( $\delta$ ). In the full version of the paper we show that the second part of Theorem 1 is best possible, up to the precise value of the constant  $\gamma$ .

**The algorithmic regularity lemma.** Loosely speaking, a regular partition of a graph  $G = (V, E)$  is a partition of  $(V_1, \dots, V_t)$  of  $V$  such that for “most”



index pairs  $i, j$  the bipartite subgraph spanned by  $V_i$  and  $V_j$  is quasi-random. Thus, a regular partition approximates  $G$  by quasi-random graphs. Furthermore, the number  $t$  of classes may depend on a parameter  $\varepsilon$  that rules the accuracy of the approximation, but it does *not* depend on the order of the graph  $G$  itself. Therefore, if for some class of graphs we can compute regular partitions in polynomial time, then this graph class will admit polynomial time algorithms for quite a few problems that are NP-hard in general.

In the sequel we introduce a new concept of regular partitions that takes into account the degree distribution of the graph. If  $G = (V, E)$  is a graph and  $A, B \subset V$  are disjoint, then the *relative density* of  $(A, B)$  in  $G$  is  $\rho(A, B) = \frac{e(A, B)}{\text{vol}(A)\text{vol}(B)}$ . Further, we say that the pair  $(A, B)$  is  $\varepsilon$ -*volume regular* if for all  $X \subset A, Y \subset B$  satisfying  $\text{vol}(X) \geq \varepsilon \text{vol}(A), \text{vol}(Y) \geq \varepsilon \text{vol}(B)$  we have

$$|e(X, Y) - \rho(A, B)\text{vol}(X)\text{vol}(Y)| \leq \varepsilon \cdot \text{vol}(A)\text{vol}(B)/\text{vol}(V), \tag{2}$$

where  $e(X, Y)$  denotes the number of  $X$ - $Y$ -edges in  $G$ . This condition essentially means that the bipartite graph spanned by  $A$  and  $B$  is quasi-random, given the degree distribution of  $G$ . Indeed, in a random graph the proportion of edges between  $X$  and  $Y$  should be proportional to both  $\text{vol}(X)$  and  $\text{vol}(Y)$ , and hence to  $\text{vol}(X)\text{vol}(Y)$ . Moreover,  $\rho(A, B)$  measures the overall density of  $(A, B)$ .

Finally, we state a condition that ensures the existence of regular partitions. While *every* dense graph  $G$  (of volume  $\text{vol}(V) = \Omega(|V|^2)$ ) admits a regular partition, such partitions do not necessarily exist for sparse graphs, the basic obstacle being extremely “dense spots”. To rule out such dense spots, we say that a graph  $G$  is  $(C, \eta)$ -*bounded* if for all  $X, Y \subset V$  with  $\text{vol}(X \cup Y) \geq \eta \text{vol}(V)$  we have  $\rho(X, Y)\text{vol}(V) \leq C$ .

**Theorem 2.** *For any two numbers  $C > 0$  and  $\varepsilon > 0$  there exist  $\eta > 0$  and  $n_0 > 0$  such that for all  $n > n_0$  the following holds. If  $G = (V, E)$  is a  $(C, \eta)$ -bounded graph on  $n$  vertices such that  $\text{vol}(V) \geq \eta^{-1}n$ , then there is a partition  $\mathcal{P} = \{V_i: 0 \leq i \leq t\}$  of  $V$  that enjoys the following two properties.*

**REG1.** *For all  $1 \leq i \leq t$  we have  $\eta \text{vol}(V) \leq \text{vol}(V_i) \leq \varepsilon \text{vol}(V)$ , and  $\text{vol}(V_0) \leq \varepsilon \text{vol}(V)$ .*

**REG2.** *Let  $\mathcal{L}$  be the set of all pairs  $(i, j) \in \{1, \dots, t\}^2$  such that the pair  $(V_i, V_j)$  is not  $\varepsilon$ -volume-regular. Then  $\sum_{(i, j) \in \mathcal{L}} \text{vol}(V_i)\text{vol}(V_j) \leq \varepsilon \text{vol}^2(G)$ .*

*Furthermore, for fixed  $C > 0$  and  $\varepsilon > 0$  such a partition  $\mathcal{P}$  of  $V$  can be computed in time polynomial in  $n$ . More precisely, the running time is  $O(\text{vol}(V) + \text{ApxCutNorm}(n))$ , where  $\text{ApxCutNorm}(n)$  is the running time of the algorithm from Theorem 5 for an  $n \times n$  matrix, which can be solved via semidefinite programming.*

Theorem 2 can be applied to the MAX CUT problem. While approximating MAX CUT within a ratio better than  $\frac{16}{17}$  is NP-hard on general graphs [14,19], the following theorem provides a polynomial time approximation scheme for  $(C, \eta)$ -bounded graphs.

**Theorem 3.** *For any  $\delta > 0$  and  $C > 0$  there exist two numbers  $\eta > 0$ ,  $n_0$  and a polynomial time algorithm `ApxMaxCut` such that for all  $n > n_0$  the following is true. If  $G = (V, E)$  is a  $(C, \eta)$ -bounded graph on  $n$  vertices and  $\text{vol}(V) > \eta^{-1}|V|$ , then `ApxMaxCut`( $G$ ) outputs a cut  $(S, \bar{S})$  of  $G$  that approximates the maximum cut within a factor of  $1 - \delta$ .*

The details of the proof of Theorem 3 will be given in the full version of the paper. The proof follows the ideas of Frieze and Kannan from [10], where the corresponding result for dense graphs was obtained.

## 2 Related Work

**Quasi-random graphs.** Quasi-random graphs with general degree distributions were first studied by Chung and Graham [7]. They considered the properties  $\text{Disc}(\varepsilon)$  and  $\text{Eig}(\delta)$ , and a number of further related ones (e.g., concerning weighted cycles). Chung and Graham observed that  $\text{Eig}(\delta)$  implies  $\text{Disc}(\varepsilon)$ , and that the converse is true in the case of *dense* graphs (i.e.,  $\text{vol}(V) = \Omega(|V|^2)$ ).

Regarding the step from  $\text{Disc}(\varepsilon)$  to  $\text{Eig}(\delta)$ , Butler [6] proved that any graph  $G$  such that for all sets  $X, Y \subset V$  the bound

$$|e(X, Y) - \text{vol}(X)\text{vol}(Y)/\text{vol}(V)| \leq \varepsilon \sqrt{\text{vol}(X)\text{vol}(Y)} \tag{3}$$

holds, satisfies  $\text{Eig}(O(\varepsilon(1 - \ln \varepsilon)))$ . The proof builds heavily on the work of Bilu and Linial [5], who derived a similar result for regular graphs.

Butler’s result relates to the second part of Theorem 1 as follows. The r.h.s. of (3) refers to the volumes of the sets  $X, Y$ , and may thus be significantly smaller than  $\varepsilon \text{vol}(V)$ . By contrast, the second part of Theorem 1 just requires that the “original” discrepancy condition  $\text{Disc}(\delta)$  is true, i.e., we just need to bound  $|e(S) - \frac{1}{2}\text{vol}(S)^2/\text{vol}(V)|$  in terms of the *total* volume  $\text{vol}(V)$ . Thus, Theorem 1 requires a considerably weaker assumption. Indeed, providing a characterization of  $\text{Disc}(\delta)$  in terms of eigenvalues, Theorem 1 answers a question posed by Chung and Graham [7,8]. Furthermore, relying on Grothendieck’s inequality and SDP duality, the proof of Theorem 1 employs quite different techniques than those used in [5,6].

In the present work we consider a concept of quasi-randomness that takes into account the graph’s degree sequence. Other concepts that do not refer to the degree sequence (and are therefore restricted to approximately regular graphs) were studied by Chung, Graham and Wilson [9] (dense graphs) and by Chung and Graham [8] (sparse graphs). Also in this setting it has been an open problem to derive eigenvalue separation from low discrepancy, and concerning this simpler concept of quasi-randomness, our techniques yield a similar result as Theorem 1 as well (details omitted).

**Regular partitions.** Szemerédi’s original regularity lemma [18] shows that any *dense* graph  $G = (V, E)$  (with  $|E| = \Omega(|V|^2)$ ) can be partitioned into a bounded number of sets  $V_1, \dots, V_t$  such that almost all pairs  $(V_i, V_j)$  are quasi-random. This statement has become an important tool in various areas, including

extremal graph theory and property testing. Furthermore, Alon, Duke, Lefmann, Rödl, and Yuster [3] presented an algorithmic version, and showed how this lemma can be used to provide polynomial time approximation schemes for dense instances of NP-hard problems (see also [16] for a faster algorithm). Moreover, Frieze and Kannan [10] introduced a different algorithmic regularity concept, which yields better efficiency in terms of the desired approximation guarantee.

A version of the regularity lemma that applies to sparse graphs was established independently by Kohayakawa [15] and Rödl (unpublished). This result is of significance, e.g., in the theory of random graphs. The regularity concept of Kohayakawa and Rödl is related to the notion of quasi-randomness from [8] and shows that any graph that satisfies a certain boundedness condition has a regular partition.

In comparison to the Kohayakawa-Rödl regularity lemma, the new aspect of Theorem 2 is that it takes into account the graph’s degree distribution. Therefore, Theorem 2 applies to graphs with very irregular degree distributions, which were not covered by prior versions of the sparse regularity lemma. Further, Theorem 2 yields an efficient algorithm for computing a regular partition (see e.g. [11] for a non-polynomial time algorithm in the sparse setting). To achieve this algorithmic result, we build upon the algorithmic version of Grothendieck’s inequality due to Alon and Naor [4]. Besides, our approach can easily be modified to obtain a polynomial time algorithm for computing a regular partition in the sense of Kohayakawa and Rödl.

### 3 Preliminaries

If  $S \subset V$  is a subset of some set  $V$ , then we let  $\mathbf{1}_S \in \mathbf{R}^V$  denote the vector whose entries are 1 on the entries corresponding to elements of  $S$ , and 0 otherwise. Moreover, if  $A = (a_{vw})_{v,w \in V}$  is a matrix, then  $A_S = (a_{vw})_{v,w \in S}$  denotes the minor of  $A$  induced on  $S \times S$ . In addition, if  $\xi = (\xi_v)_{v \in V}$  is a vector, then  $\text{diag}(\xi)$  signifies the  $V \times V$  matrix with diagonal  $\xi$  and off-diagonal entries equal to 0. Further, for a vector  $\xi \in \mathbf{R}^V$  we let  $\|\xi\|$  signify the  $\ell_2$ -norm, and for a matrix we let  $\|M\| = \sup_{0 \neq \xi \in \mathbf{R}^V} \frac{\|M\xi\|}{\|\xi\|}$  denote the spectral norm. If  $M$  is symmetric, then  $\lambda_{\max}(M)$  denotes the largest eigenvalue of  $M$ .

An important ingredient to our proofs and algorithms is Grothendieck’s inequality. Let  $M = (m_{ij})_{i,j \in \mathcal{I}}$  be a matrix. Then the *cut-norm* of  $M$  is  $\|M\|_{\text{cut}} = \max_{I,J \subset \mathcal{I}} \left| \sum_{i \in I, j \in J} m_{ij} \right|$ . In addition, consider the following optimization problem:

$$\text{SDP}(M) = \max \sum_{i,j \in \mathcal{I}} m_{ij} \langle x_i, y_j \rangle \text{ s.t. } \|x_i\| = \|y_i\| = 1, x_i, y_i \in \mathbf{R}^{\mathcal{I}}.$$

Then  $\text{SDP}(M)$  can be reformulated as a *linear* optimization problem over the cone of positive semidefinite  $2|\mathcal{I}| \times 2|\mathcal{I}|$  matrices, i.e., as a semidefinite program (cf. Alizadeh [2]). Hence, an optimal solution to  $\text{SDP}(M)$  can be approximated within any numerical precision, e.g., via the ellipsoid method [13]. Grothendieck [12] proved the following relation between  $\text{SDP}(M)$  and  $\|M\|_{\text{cut}}$ .

**Theorem 4.** *There is a constant  $\theta > 1$  such that for all matrices  $M$  we have  $\|M\|_{\text{cut}} \leq \text{SDP}(M) \leq \theta \cdot \|M\|_{\text{cut}}$ .*

The best current bounds on the above constant are  $\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2 \ln(1+\sqrt{2})}$  [12][17]. Furthermore, by applying an appropriate rounding procedure to a near-optimal solution to  $\text{SDP}(M)$ , Alon and Naor [4] obtained the following algorithmic result.

**Theorem 5.** *There exist  $\theta' > 0$  and a polynomial time algorithm  $\text{ApxCutNorm}$  that computes on input  $M$  sets  $I, J \subset \mathcal{I}$  such that  $\theta' \cdot \|M\|_{\text{cut}} \leq |\sum_{i \in I, j \in J} m_{ij}|$ .*

Alon and Naor presented a randomized algorithm that guarantees an approximation ratio  $\theta' > 0.56$ , and a deterministic one with  $\theta' \geq 0.03$ .

### 4 Quasi-randomness: Proof of Theorem 1

The proof of the first part of Theorem 1 is similar to the proof given in [7 Section 4]. Thus, we focus on the second implication, and hence assume that  $G = (V, E)$  is a graph that has  $\text{Disc}(\gamma\varepsilon^3)$ , where  $\gamma > 0$  signifies some small enough constant (e.g.,  $\gamma = (6400\theta)^{-1}$  suffices for the proof below). Moreover, we let  $d_v$  denote the degree of  $v \in V$ ,  $n = |V|$ , and  $\bar{d} = n^{-1} \sum_{v \in V} d_v$ . In addition, we introduce a further property.

**Cut( $\varepsilon$ ):** We say  $G$  has  $\text{Cut}(\varepsilon)$ , if the matrix  $M = (m_{vw})_{v,w \in V}$  with entries  $m_{vw} = \frac{d_v d_w}{\text{vol}(V)} - e(\{v\}, \{w\})$  has cut norm  $\|M\|_{\text{cut}} < \varepsilon \cdot \text{vol}(V)$ , where  $e(\{v\}, \{w\}) = 1$  if  $\{v, w\} \in E$  and 0 otherwise.

Since for any  $S \subset V$  we have  $\langle M \mathbf{1}_S, \mathbf{1}_S \rangle = \frac{\text{vol}(S)^2}{\text{vol}(V)} - 2e(S)$ , one can easily derive the following.

**Proposition 6.** *Each graph that has  $\text{Disc}(0.01\delta)$  enjoys  $\text{Cut}(\delta)$ .*

To show that  $\text{Disc}(\gamma\varepsilon^3)$  implies  $\text{ess-Eig}(\varepsilon)$ , we proceed as follows. By Proposition 6,  $\text{Disc}(\gamma\varepsilon^3)$  implies  $\text{Cut}(100\gamma\varepsilon^3)$ . Moreover, if  $G$  satisfies  $\text{Cut}(100\gamma\varepsilon^3)$ , then Theorem 4 entails that not only the cut norm of  $M$  is small, but even the semidefinite relaxation  $\text{SDP}(M)$  satisfies  $\text{SDP}(M) < \beta\varepsilon^3 \text{vol}(V)$ , for some  $\beta$  with  $0 < \beta \leq 100\theta\gamma$ . This bound on  $\text{SDP}(M)$  can be rephrased in terms of an eigenvalue minimization problem for a matrix closely related to  $M$ . More precisely, using the duality theorem for semidefinite programs, we can infer the following.

**Lemma 7.** *For any symmetric  $n \times n$  matrix  $Q$  we have*

$$\text{SDP}(Q) = n \cdot \min_{z \in \mathbb{R}^n, z \perp \mathbf{1}} \lambda_{\max} \left[ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes Q - \text{diag} \begin{pmatrix} z \\ z \end{pmatrix} \right].$$

Let  $D = \text{diag}(d_v)_{v \in V}$ . Then Lemma 7 entails the following.

**Lemma 8.** *Suppose that  $\text{SDP}(M) < \beta\varepsilon^3 \text{vol}(V)$  for some  $\beta$ ,  $0 < \beta < 1/64$ . Then there exists a subset  $W \subset V$  of volume  $\text{vol}(W) \geq (1 - \varepsilon) \cdot \text{vol}(V)$  such that the matrix  $\mathcal{M} = D^{-\frac{1}{2}} M D^{-\frac{1}{2}}$  satisfies  $\|\mathcal{M}_W\| < \varepsilon$ .*

*Proof.* Let  $U = \{v \in V : d_v > \beta^{\frac{1}{3}}\varepsilon\bar{d}\}$ . Then

$$\text{vol}(V \setminus U) \leq \beta^{\frac{1}{3}}\varepsilon\bar{d}|V \setminus U| \leq \varepsilon\text{vol}(V)/2. \tag{4}$$

Since  $\text{SDP}(M_U) \leq \text{SDP}(M)$ , Lemma 7 entails that there is a vector  $\mathbf{1} \perp z \in \mathbf{R}^U$  such that  $\lambda_{\max} \left[ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes M_U - \text{diag} \begin{pmatrix} z \\ z \end{pmatrix} \right] < \beta\varepsilon^3\bar{d}$ . Hence, setting  $y = D_U^{-1}z$ , we obtain

$$\lambda_{\max} \left[ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \mathcal{M}_U - \text{diag} \begin{pmatrix} y \\ y \end{pmatrix} \right] < \beta^{\frac{2}{3}}\varepsilon^2, \tag{5}$$

because all entries of the diagonal matrix  $D_U$  exceed  $\beta^{\frac{1}{3}}\varepsilon\bar{d}$ . Moreover, as  $z \perp \mathbf{1}$ , we have

$$y \perp D_U\mathbf{1}. \tag{6}$$

Now, let  $W = \{v \in U : |y_v| < \beta^{\frac{1}{3}}\varepsilon\}$  consist of all vertices  $v$  on which the ‘‘correcting vector’’  $y$  is small. Since on  $W$  all entries of the diagonal matrix  $\text{diag} \begin{pmatrix} y \\ y \end{pmatrix}$  are smaller than  $\beta^{\frac{1}{3}}\varepsilon$  in absolute value, (5) yields

$$\lambda_{\max} \left[ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \mathcal{M}_W \right] < \beta^{\frac{1}{3}}\varepsilon + \beta^{\frac{2}{3}}\varepsilon^2 \leq 2\beta^{\frac{1}{3}}\varepsilon; \tag{7}$$

in other words, on  $W$  the effect of  $y$  is negligible. Further, (7) entails that  $\|\mathcal{M}_W\| \leq 2\beta^{\frac{1}{3}}\varepsilon < \varepsilon$ .

Finally, we need to show that  $\text{vol}(W)$  is large. To this end, we consider the set  $S = \{v \in U : y_v < 0\}$  and let  $\zeta = D_U^{\frac{1}{2}}\mathbf{1}_S$ . Thus, for each  $v \in U$  the entry  $\zeta_v$  equals  $d_v^{\frac{1}{2}}$  if  $y_v < 0$ , while  $\zeta_v = 0$  if  $y_v \geq 0$ , so that  $\|\zeta\|^2 = \text{vol}(S)$ . Hence, (5) yields that

$$\begin{aligned} 2\beta^{\frac{2}{3}}\varepsilon^2\text{vol}(S) &= 2\beta^{\frac{2}{3}}\varepsilon^2\|\zeta\|^2 \geq \left\langle \left[ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \mathcal{M}_U - \text{diag} \begin{pmatrix} y \\ y \end{pmatrix} \right] \cdot \begin{pmatrix} \zeta \\ \zeta \end{pmatrix}, \begin{pmatrix} \zeta \\ \zeta \end{pmatrix} \right\rangle \\ &= 2 \langle \mathcal{M}_U\zeta, \zeta \rangle - 2 \sum_{v \in S} d_v y_v = 2 \langle M_U\mathbf{1}_S, \mathbf{1}_S \rangle - 2 \sum_{v \in S} d_v y_v. \end{aligned} \tag{8}$$

Furthermore, as  $\text{SDP}(M_U) \leq \text{SDP}(M) \leq \beta\varepsilon^3\text{vol}(V)$ , Theorem 4 entails that  $\langle M_U\mathbf{1}_S, \mathbf{1}_S \rangle \leq \|M_U\|_{\text{cut}} \leq \beta\varepsilon^3\text{vol}(V)$ . Plugging this bound into (8) and recalling that  $y_v < 0$  for all  $v \in S$ , we conclude that

$$\sum_{v \in S} d_v |y_v| \leq \beta^{\frac{2}{3}}\varepsilon^2\text{vol}(S) + \beta\varepsilon^3\text{vol}(V) \leq 2\beta^{\frac{2}{3}}\varepsilon^2\text{vol}(V). \tag{9}$$

Hence, (6) entails that actually  $\sum_{v \in U} d_v |y_v| \leq 4\beta^{\frac{2}{3}}\varepsilon^2\text{vol}(V)$ . As  $|y_v| \geq \beta^{\frac{1}{3}}\varepsilon$  for all  $v \in U \setminus W$ , we obtain  $\text{vol}(U \setminus W) \leq 4\beta^{\frac{1}{3}}\varepsilon\text{vol}(V) < \frac{1}{2}\varepsilon\text{vol}(V)$ . Thus, (4) yields  $\text{vol}(V \setminus W) < \varepsilon\text{vol}(V)$ , as desired.  $\square$

Finally, setting  $\gamma = (6400\theta)^{-1}$  and combining Theorem 4, Proposition 6, and Lemma 8, we conclude if  $G$  has  $\text{Disc}(\gamma\varepsilon^3)$ , then there is a set  $W$  such that

$\text{vol}(W) \geq (1 - \varepsilon)\text{vol}(V)$  and  $\|\mathcal{M}_W\| < \varepsilon$ . As  $\mathcal{M}$  is closely related to the normalized Laplacian  $L(G)$ , one can infer via elementary linear algebra that the minor  $L(G)_W$  corresponding to  $W$  satisfies  $1 - \varepsilon \leq \lambda_2(L(G)_W) \leq \lambda_{|W|}(L(G)_W) \leq 1 + \varepsilon$ , whence  $G$  has  $\text{ess-Eig}(\varepsilon)$ .

## 5 The Algorithmic Regularity Lemma

In this section we present a polynomial time algorithm **Regularize** that computes for a given graph  $G = (V, E)$  a partition satisfying **REG1** and **REG2**, provided that  $G$  satisfies the assumptions of Theorem 2. In particular, this will show that such a partition exists. We will outline **Regularize** in Section 5.1. The crucial ingredient is a subroutine **Witness** for checking whether a given pair  $(A, B)$  of subsets of  $V$  is  $\varepsilon$ -volume regular. This subroutine is the content of Section 5.2.

Throughout this section, we let  $\varepsilon > 0$  be an arbitrarily small but fixed and  $C > 0$  an arbitrarily large but fixed number. In addition, we define a sequence  $(t_k)_{k \geq 1}$  by letting  $t_1 = \lceil 2/\varepsilon \rceil$  and  $t_{k+1} = t_k 2^{t_k}$ . Let  $k^* = \lceil C\varepsilon^{-3} \rceil$ ,  $\eta = t_{k^*}^{-6} \varepsilon^{-8k^*}$ , and choose  $n_0 > 0$  big enough.

We always assume that  $G = (V, E)$  is a graph on  $n = |V| > n_0$  vertices that is  $(C, \eta)$ -bounded, and that  $\text{vol}(V) \geq \eta^{-1}n$ .

### 5.1 The Algorithm Regularize

In order to compute the desired regular partition of its input graph  $G$ , the algorithm **Regularize** proceeds as follows. In its first step, **Regularize** computes any initial partition  $\mathcal{P}^1 = \{V_i^1 : 0 \leq i \leq s_1\}$  such that each class  $V_i$  ( $1 \leq i \leq s_1$ ) has a decent volume.

**Algorithm 9.** **Regularize**( $G$ )

*Input:* A graph  $G = (V, E)$ . *Output:* A partition of  $V$ .

1. Compute an initial partition  $\mathcal{P}^1 = \{V_i^1 : 0 \leq i \leq s_1\}$  such that  $\frac{1}{4}\varepsilon\text{vol}(V) \leq \text{vol}(V_i^1) \leq \frac{3}{4}\varepsilon\text{vol}(V)$  for all  $1 \leq i \leq s_1$ ; thus,  $s_1 \leq 4\varepsilon^{-1}$ . Set  $V_0^1 = \emptyset$ .

Then, in the subsequent steps, **Regularize** computes a sequence  $\mathcal{P}^k$  of partitions such that  $\mathcal{P}^{k+1}$  is a “more regular” refinement of  $\mathcal{P}^k$  ( $k \geq 1$ ). As soon as **Regularize** can verify that  $\mathcal{P}^k$  satisfies both **REG1** and **REG2**, the algorithm stops.

To check whether the current partition  $\mathcal{P}^k = \{V_i^k : 1 \leq i \leq s_1\}$  satisfies **REG2**, **Regularize** employs a subroutine **Witness**. Given a pair  $(V_i^k, V_j^k)$ , **Witness** tries to check whether  $(V_i^k, V_j^k)$  is  $\varepsilon$ -volume-regular.

**Proposition 10.** *There is a polynomial time algorithm **Witness** that satisfies the following. Let  $A, B \subset V$  be disjoint.*

1. If  $\text{Witness}(G, A, B)$  answers “yes”, then the pair  $(A, B)$  is  $\varepsilon$ -volume regular.

2. On the other hand, if the answer is “no”, then  $(A, B)$  is not  $\varepsilon/200$ -volume regular. In this case **Witness** outputs a pair  $(X^*, Y^*)$  of subsets  $X^* \subset A, Y^* \subset B$  such that  $\text{vol}(X^*) \geq \frac{\varepsilon}{200}\text{vol}(A), \text{vol}(Y^*) \geq \frac{\varepsilon}{200}\text{vol}(B)$ , and  $|e(X^*, Y^*) - \varrho(A, B)\text{vol}(X^*)\text{vol}(Y^*)| > \frac{\varepsilon\text{vol}(A)\text{vol}(B)}{200\text{vol}(V)}$ .

We call a pair  $(X^*, Y^*)$  as in 2. an  $\frac{\varepsilon}{200}$ -witness for  $(A, B)$ .

By applying **Witness** to each pair  $(V_i^k, V_j^k)$  of the partition  $\mathcal{P}^k$ , **Regularize** can single out a set  $\mathcal{L}^k$  such that all pairs  $V_i, V_j$  with  $(i, j) \notin \mathcal{L}^k$  are  $\varepsilon$ -volume regular. Hence, if  $\sum_{(i,j) \in \mathcal{L}^k} \text{vol}(V_i^k)\text{vol}(V_j^k) < \varepsilon\text{vol}(V)^2$ , then  $\mathcal{P}^k$  satisfies **REG2**. As we will see below that by construction  $\mathcal{P}^k$  satisfies **REG1** for all  $k$ , in this case  $\mathcal{P}^k$  is a feasible regular partition, whence **Regularize** stops.

2. For  $k = 1, 2, 3, \dots, k^*$  do
3. Initially, let  $\mathcal{L}^k = \emptyset$ .  
For each pair  $(V_i^k, V_j^k)$  ( $i < j$ ) of classes of the previously partition  $\mathcal{P}^k$
4. call the procedure **Witness** $(G, V_i^k, V_j^k, \varepsilon)$ .  
If it answers “no” and hence outputs an  $\frac{\varepsilon}{200}$ -witness  $(X_{ij}^k, X_{ji}^k)$  for  $(V_i^k, V_j^k)$ , then add  $(i, j)$  to  $\mathcal{L}^k$ .
5. If  $\sum_{(i,j) \in \mathcal{L}^k} \text{vol}(V_i^k)\text{vol}(V_j^k) < \varepsilon\text{vol}(V)^2$ , then output the partition  $\mathcal{P}^k$  and halt.

If Step 5 does not halt, **Regularize** constructs a refinement  $\mathcal{P}^{k+1}$  of  $\mathcal{P}^k$ . To this end, the algorithm decomposes each class  $V_i^k$  of  $\mathcal{P}^k$  into up to  $2^{s_k}$  pieces. Consider the sets  $X_{ij}$  with  $(i, j) \in \mathcal{L}^k$  and define an equivalence relation  $\equiv_i^k$  on  $V_i$  by letting  $u \equiv_i^k v$  iff for all  $j$  such that  $(i, j) \in \mathcal{L}^k$  it is true that  $u \in X_{ij} \leftrightarrow v \in X_{ij}$ . Thus, the equivalence classes of  $\equiv_i^k$  are the regions of the Venn diagram of the sets  $V_i$  and  $X_{ij}$  with  $(i, j) \in \mathcal{L}^k$ . Then **Regularize** obtains  $\mathcal{P}^{k+1}$  as follows.

6. Let  $\mathcal{C}^k$  be the set of all equivalence classes of the relations  $\equiv_i^k$  ( $1 \leq i \leq s_k$ ). Moreover, let  $\mathcal{C}_*^k = \{V_1^{k+1}, \dots, V_{s_{k+1}}^{k+1}\}$  be the set of all classes  $W \in \mathcal{C}$  such that  $\text{vol}(W) > \varepsilon^{4(k+1)}\text{vol}(V)/(15t_{k+1}^3)$ . Finally, let  $V_0^{k+1} = V_0^k \cup \bigcup_{W \in \mathcal{C}^k \setminus \mathcal{C}_*^k} W$ , and set  $\mathcal{P}^{k+1} = \{V_i^{k+1} : 0 \leq i \leq s_{k+1}\}$ .

Since for each  $i$  there are at most  $s_k$  indices  $j$  such that  $(i, j) \in \mathcal{L}^k$ , in  $\mathcal{P}^{k+1}$  every class  $V_i^k$  gets split into at most  $2^{s_k}$  pieces. Hence,  $s_{k+1} \leq s_k 2^{s_k}$ . Thus, as  $s_1 \leq t_1$ , we conclude that  $s_k \leq t_k$  for all  $k$ . Therefore, our choice of  $\eta$  ensures that  $\text{vol}(V_i^{k+1}) \geq \eta\text{vol}(V)$  for all  $1 \leq i \leq s_{k+1}$  (because Step 6 puts all equivalence classes  $W \in \mathcal{C}^k$  of “extremely small” volume into the exceptional class). Moreover, it is easily seen that  $\text{vol}(V_0^{k+1}) \leq \varepsilon(1 - 2^{k+2})\text{vol}(V)$ . In effect,  $\mathcal{P}^{k+1}$  satisfies **REG1**.

Thus, to complete the proof of Theorem 2 it just remains to show that **Regularize** will actually succeed and output a partition  $\mathcal{P}^k$  for some  $k \leq k^*$ . To show this, we define the *index* of a partition  $\mathcal{P} = \{V_i : 0 \leq i \leq s\}$  as

$$\text{ind}(\mathcal{P}) = \sum_{1 \leq i < j \leq s} \varrho(V_i, V_j)^2 \text{vol}(V_i)\text{vol}(V_j) = \sum_{1 \leq i < j \leq s} \frac{e(V_i, V_j)^2}{\text{vol}(V_i)\text{vol}(V_j)}.$$

Note that we do *not* take into account the (exceptional) class  $V_0$  here. Using the boundedness-condition, we derive the following.

**Proposition 11.** *If  $G = (V, E)$  is  $(C, \eta)$ -bounded and  $\mathcal{P} = \{V_i : 0 \leq 1 \leq t\}$  is a partition of  $V$  with  $\text{vol}(V_i) \geq \eta \text{vol}(V)$  for all  $i \in \{1, \dots, t\}$ , then  $\text{ind}(\mathcal{P}) \leq C$ .*

Lemma 11 entails that  $\text{ind}(\mathcal{P}^k) \leq C$  for all  $k$ . In addition, since **Regularize** obtains  $\mathcal{P}^{k+1}$  by refining  $\mathcal{P}^k$  according to the witnesses of irregularity computed by **Witness**, the index of  $\mathcal{P}^{k+1}$  is actually considerably larger than the index of  $\mathcal{P}^k$ . More precisely, the following is true.

**Lemma 12.**  $\sum_{(i,j) \in \mathcal{L}^k} \text{vol}(V_i^k) \text{vol}(V_j^k) \geq \varepsilon \text{vol}(V)^2 \Rightarrow \text{ind}(\mathcal{P}^{k+1}) \geq \text{ind}(\mathcal{P}^k) + \frac{\varepsilon^3}{8}$ .

Since the index of the initial partition  $\mathcal{P}^1$  is non-negative, Lemmas 11 and 12 readily imply that **Regularize** will terminate and output a feasible partition  $\mathcal{P}^k$  for some  $k < k^*$ .

Finally, we point out that the overall running time of **Regularize** is polynomial. For the running time of Steps 1–3 and 5–6 is  $O(\text{vol}(V))$ , and the running time of Step 4 is polynomial due to Proposition 10.

### 5.2 The Procedure Witness

The subroutine **Witness** for Proposition 10 employs the algorithm **ApxCutNorm** from Theorem 5 for approximating the cut norm as follows.

**Algorithm 13.** **Witness**( $G, A, B$ )

*Input:* A graph  $G = (V, E)$ , disjoint sets  $A, B \subset V$ , and a number  $\varepsilon > 0$ .

*Output:* A partition of  $V$ .

1. Set up a matrix  $M = (m_{vw})_{(v,w) \in A \times B}$  with entries  $m_{vw} = 1 - \varrho(A, B) d_v d_w$  if  $v, w$  are adjacent in  $G$ , and  $m_{vw} = -\varrho(A, B) d_v d_w$  otherwise. Call **ApxCutNorm**( $M$ ) to compute sets  $X \subset A, Y \subset B$  such that  $|\langle M \mathbf{1}_X, \mathbf{1}_Y \rangle| \geq \frac{3}{100} \|M\|_{\text{cut}}$ .
2. If  $|\langle M \mathbf{1}_X, \mathbf{1}_Y \rangle| < 3\varepsilon/100$ , then return “yes”.
3. Otherwise, pick  $X' \subset A \setminus X$  of volume  $\frac{3\varepsilon}{100} \text{vol}(A) \leq \text{vol}(X') \leq \frac{4\varepsilon}{100} \text{vol}(A)$ .
  - If  $\text{vol}(X) \geq \frac{3\varepsilon}{100} \text{vol}(A)$ , then let  $X^* = X$ .
  - If  $\text{vol}(X) < \frac{3\varepsilon}{100} \text{vol}(A)$  and  $|e(X', Y) - \varrho(A, B) \text{vol}(X') \text{vol}(Y)| > \frac{\varepsilon \text{vol}(A) \text{vol}(B)}{100 \text{vol}(V)}$ , set  $X^* = X'$ .
  - Otherwise, set  $X^* = X \cup X'$ .
4. Pick a further set  $Y' \subset B \setminus Y$  of volume  $\frac{\varepsilon}{200} \text{vol}(B) \leq \text{vol}(Y') \leq \frac{2\varepsilon}{300} \text{vol}(B)$ .
  - If  $\text{vol}(Y) \geq \frac{\varepsilon}{200} \text{vol}(B)$ , then let  $Y^* = Y$ .
  - If  $\text{vol}(Y) < \frac{\varepsilon}{200} \text{vol}(B)$  and  $|e(X^*, Y') - \varrho(A, B) \text{vol}(X^*) \text{vol}(Y')| > \frac{\varepsilon \text{vol}(A) \text{vol}(B)}{200 \text{vol}(V)}$ , let  $Y^* = Y'$ .
  - Otherwise, set  $Y^* = Y \cup Y'$ .
5. Answer “no” and output  $(X^*, Y^*)$  as an  $\varepsilon/8$ -witness.

Given the graph  $G$  along with two disjoint sets  $A, B \subset V$ , **Witness** sets up a matrix  $M$ . The crucial property of  $M$  is that for any two subsets  $S \subset A$  and  $T \subset B$  we have  $\langle M \mathbf{1}_S, \mathbf{1}_T \rangle = e(S, T) - \varrho(A, B) \text{vol}(S) \text{vol}(T)$ . Therefore, if  $\|M\|_{\text{cut}} \leq \varepsilon \text{vol}(A) \text{vol}(B) / \text{vol}(V)$ , then the pair  $(A, B)$  is  $\varepsilon$ -volume regular. Hence, in order to find out whether  $(A, B)$  is  $\varepsilon$ -volume regular, **Witness** employs the algorithm **ApxCutNorm** to approximate  $\|M\|_{\text{cut}}$ . If Step 2 of **Witness** answers “yes”, then



$(A, B)$  is  $\varepsilon$ -volume regular, because **ApxCutNorm** achieves an approximation ratio  $> \frac{3}{100}$  by Theorem 5.

On the other hand, if **ApxCutNorm** yields sets  $X, Y$  such that  $|\langle M\mathbf{1}_X, \mathbf{1}_Y \rangle| > \frac{3\varepsilon\text{vol}(A)\text{vol}(B)}{100\text{vol}(V)}$ , then **Witness** constructs an  $\varepsilon/200$ -witness for  $(A, B)$ . Indeed, if the volumes of  $X$  and  $Y$  are “large enough” – say,  $\text{vol}(X) \geq \frac{\varepsilon}{200}\text{vol}(A)$  and  $\text{vol}(Y) \geq \frac{\varepsilon}{200}\text{vol}(B)$  – then  $(X, Y)$  actually is an  $\varepsilon/200$ -witness. However, as **ApxCutNorm** does not guarantee any lower bound on  $\text{vol}(X)$ ,  $\text{vol}(Y)$ , Steps 3–5 try to enlarge the sets  $X, Y$  a little if their volume is too small. Finally, it is straightforward to verify that this procedure yields an  $\varepsilon/200$ -witness  $(X^*, Y^*)$ , which entails Proposition 10.

## References

1. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. *Reviews of modern physics* 74, 47–97 (2002)
2. Alizadeh, F.: Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optimization* 5, 13–51 (1995)
3. Alon, N., Duke, R.A., Rödl, V., Yuster, R.: The algorithmic aspects of the regularity lemma. *J. of Algorithms* 16, 80–109 (1994)
4. Alon, N., Naor, A.: Approximating the cut-norm via Grothendieck’s inequality. In: *Proc. 36th STOC*, pp. 72–80 (2004)
5. Bilu, Y., Linial, N.: Lifts, discrepancy and nearly optimal spectral gap. *Combinatorica (to appear)*
6. Butler, S.: On eigenvalues and the discrepancy of graphs. preprint
7. Chung, F., Graham, R.: Quasi-random graphs with given degree sequences. Preprint (2005)
8. Chung, F., Graham, R.: Sparse quasi-random graphs. *Combinatorica* 22, 217–244 (2002)
9. Chung, F., Graham, R., Wilson, R.M.: Quasi-random graphs. *Combinatorica* 9, 345–362 (1989)
10. Frieze, A., Kannan, R.: Quick approximation to matrices and applications. *Combinatorica* 19, 175–200 (1999)
11. Gerke, S., Steger, A.: A characterization for sparse  $\varepsilon$ -regular pairs. *The Electronic J. Combinatorics* 14, R4, p. 12 (2007)
12. Grothendieck, A.: Résumé de la théorie métrique des produits tensoriels topologiques. *Bol. Soc. Mat. Sao Paulo* 8, 1–79 (1953)
13. Grötschel, M., Lovász, L., Schrijver, A.: *Geometric algorithms and combinatorial optimization*. Springer, Heidelberg (1988)
14. Hastad, J.: Some optimal inapproximability results. *J. of the ACM* 48, 798–859 (2001)
15. Kohayakawa, Y.: Szemerédi’s regularity lemma for sparse graphs. In: Cucker, F., Shub, M. (eds.) *Foundations of computational mathematics*, pp. 216–230 (1997)
16. Kohayakawa, Y., Rödl, V., Thoma, L.: An optimal algorithm for checking regularity. *SIAM J. Comput.* 32, 1210–1235 (2003)
17. Krivine, J.L.: Sur la constante de Grothendieck. *C. R. Acad. Sci. Paris Ser. A-B* 284, 445–446 (1977)
18. Szemerédi, E.: Regular partitions of graphs. *Problèmes Combinatoires et Théorie des Graphes Colloques Internationaux CNRS* 260, 399–401 (1978)
19. Trevisan, L., Sorkin, G., Sudan, M., Williamson, D.: Gadgets, approximation, and linear programming. *SIAM J. Computing* 29, 2074–2097 (2000)

# Complexity of the Cover Polynomial

Markus Bläser and Holger Dell

Computational Complexity Group  
Saarland University, Germany  
{mblaeser,hdell}@cs.uni-sb.de

**Abstract.** The cover polynomial introduced by Chung and Graham is a two-variate graph polynomial for directed graphs. It counts the (weighted) number of ways to cover a graph with disjoint directed cycles and paths, it is an interpolation between determinant and permanent, and it is believed to be a directed analogue of the Tutte polynomial. Jaeger, Vertigan, and Welsh showed that the Tutte polynomial is  $\#P$ -hard to evaluate at all but a few special points and curves. It turns out that the same holds for the cover polynomial: We prove that, in almost the whole plane, the problem of evaluating the cover polynomial is  $\#P$ -hard under polynomial-time Turing reductions, while only three points are easy. Our construction uses a gadget which is easier to analyze and more general than the XOR-gadget used by Valiant in his proof that the permanent is  $\#P$ -complete.

## 1 Introduction

Graph polynomials map directed or undirected graphs to polynomials in one or more variables, such that this mapping is invariant under graph isomorphisms. Probably the most famous graph polynomials are the chromatic polynomial or its generalization, the Tutte polynomial. The chromatic polynomial is the polynomial in the variable  $\lambda$  that counts the number of valid  $\lambda$ -colourings of a given undirected graph. The Tutte polynomial  $T$  in two variables  $x$  and  $y$  has interpretations from different fields of combinatorics. For example,  $T(G, 1, 1)$  is the number of spanning trees,  $T(G, 1, 2)$  is the number of spanning subgraphs of an undirected graph  $G$ , and also the number of nowhere-zero flows or the Jones polynomial of an alternating link are contained in the Tutte polynomial.

While the Tutte polynomial has been established for *undirected* graphs, the cover polynomial by Chung and Graham [1] is an analogue for the directed case. Both graph polynomials satisfy similar identities such as a contraction-deletion identity and product rule, but the exact relation between the Tutte and the cover polynomial is not yet known. The cover polynomial has connections to rook polynomials and drop polynomials, but from a complexity theoretic point of view, we tend to see it as a generalization of the permanent and the determinant of a graph. The cover polynomial of a graph is the weighted sum of all of its spanning subgraphs that consist of disjoint, directed, and simple cycles and paths. As it is the case for most graph polynomials, the cover polynomial

is of interest because it combines a variety of combinatorial problems into one generalized theoretical framework.

## 1.1 Previous Results

Jaeger, Vertigan, and Welsh [2] have shown that, except along one hyperbola and at nine special points, computing the Tutte polynomial is  $\#P$ -hard. In recent years, the complexity and approximability of the Tutte polynomial has received increasing attention: Lotz and Makowsky [3] prove that the coloured Tutte polynomial by Bollobás and Riordan [4] is complete for Valiant's algebraic complexity class VNP, Giménez and Noy [5] show that evaluating the Tutte polynomial is  $\#P$ -hard even for the rather restricted class of bicircular matroids, and Goldberg and Jerrum [6] show that the Tutte polynomial is mostly inapproximable.

A different graph invariant that seems related to the Tutte polynomial is the weighted sum of graph homomorphisms to a fixed graph  $H$ , so basically, it is the number of  $H$ -colourings. Bulatov and Grohe [7] and Dyer, Goldberg, and Paterson [8] prove that the complexity of computing this sum is  $\#P$ -hard for most graphs  $H$ .

## 1.2 Our Contribution

In this paper, we show that the problem of evaluating the cover polynomial is  $\#P$ -hard at all evaluation points except for three points where this is easy.

The big picture of the proof is as follows: We use elementary identities of the cover polynomial in order to construct simple reductions along horizontal lines. Furthermore, we establish an interesting identity along the  $y$ -axis. As it turns out, there is quite a strong connection between cover polynomial and permanent, and our construction uses an equality gadget with a similar effect as the XOR-gadget which Valiant [9] uses in his proof that the permanent is  $\#P$ -complete. Our gadget, however, is simpler to analyze and more general, in the sense that it satisfies additional properties about the number of cycles contributed by our gadget to each cycle cover.

In addition, we carry over the hardness result to the geometric cover polynomial introduced by D'Antona and Munarini [10].

## 2 Preliminaries

Let  $\mathbb{N} = \{0, 1, \dots\}$ . The graphs in this paper are directed *multigraphs*  $D = (V, E)$  with parallel edges and loops allowed. We denote by  $\mathcal{G}$  the set of all such graphs. We write  $n$  for the number of vertices, and  $m$  for the number of edges. Two graphs are called *isomorphic* if there is a bijective mapping on the vertices that transforms one graph into the other.

A *graph invariant* is a function  $f : \mathcal{G} \rightarrow F$ , mapping elements from  $\mathcal{G}$  to some set  $F$ , such that all pairs of isomorphic graphs  $G$  and  $G'$  have the same image under  $f$ . In the case that  $F$  is a polynomial ring,  $f$  is called *graph polynomial*.

**Counting Complexity Basics.** The class #P consists of all functions  $f : \{0, 1\}^* \rightarrow \mathbb{N}$  for which there is a non-deterministic polynomial-time bounded Turing machine  $M$  which has exactly  $f(x)$  accepting paths on input  $x$ . For two counting problems  $f, g : \{0, 1\}^* \rightarrow \mathbb{Q}$  not necessarily in #P, we say  $f$  *Turing-reduces* to  $g$  in polynomial time ( $f \preceq_T^P g$ ), if there is a deterministic oracle Turing machine  $M$  which computes  $f$  in polynomial time with oracle access to  $g$ . If the oracle is used only once, we say  $f$  *many-one reduces* to  $g$  ( $f \preceq_m^P g$ ), and if the oracle output is the output of the reduction, we speak of a *parsimonious* many-one reduction ( $f \preceq^P g$ ). The notions of #P-hardness and #P-completeness (under polynomial-time Turing reductions) are defined in the usual way.

**Polynomials.** Polynomials  $p(x_1, \dots, x_m)$  are elements of the polynomial ring  $\mathbb{Q}[x_1, \dots, x_m]$ , and, in this context, the variables are abstract objects. Lagrange interpolation can be used to compute the following problem in polynomial time.

*Input:* Point-value pairs  $(a_1, p_1), \dots, (a_d, p_d) \in \mathbb{Q}^2$ , encoded in binary.  
*Output:* The coefficients of the polynomial  $p(x)$  with  $\deg(p) \leq d$  and  $p(a_j) = p_j$ .

**The Cover Polynomial.** The cover polynomial basically counts a relaxed form of cycle covers, namely *path-cycle covers*. For a directed graph  $D = (V, E)$  and some subset  $C \subseteq E$ , we denote the subgraph  $(V, C)$  again by  $C$ . A path-cycle cover of  $D$  is a set  $C \subseteq E$ , such that in  $C$  every vertex  $v \in V$  has an indegree and an outdegree of at most 1. A path-cycle cover thus consists of disjoint simple paths and simple cycles. Note that also an independent vertex counts as a path, and an independent loop counts as a cycle.

By the graph invariant  $c_D(i, j)$ , we denote the number of path-cycle covers of  $D$  that have exactly  $i$  paths and  $j$  cycles. It is not hard to prove that the function  $D \mapsto c_D(i, j)$  is #P-complete (cf. [11]). The cover polynomial by Chung and Graham [1] is a graph polynomial in the variables  $x$  and  $y$ , and it is defined by the equation

$$C(D, x, y) := \sum_{i=0}^m \sum_{j=0}^m c_D(i, j) x^i y^j, \tag{1}$$

where  $x^i := x(x - 1) \dots (x - i + 1)$  denotes the falling factorial.

Writing  $i(C)$  and  $j(C)$  for the number of paths and cycles of a path-cycle cover  $C$ , the cover polynomial is the weighted sum over all covers of  $D$ :

$$C(D, x, y) = \sum_{\substack{\text{path-cycle} \\ \text{cover } C}} x^{i(C)} y^{j(C)}.$$

### 3 Overview

The problem of evaluating the cover polynomial, written  $C(a, b)$ , is parameterized by the coordinates  $(a, b)$ . Formally,  $C(a, b)$  is the function from  $\mathcal{G} \rightarrow \mathbb{Q}$  with  $D \mapsto C(D, a, b)$  where we assume graphs and rationals to be represented explicitly in a standard way. Our main theorem is the following.

**Main Theorem.** *Let  $(a, b) \in \mathbb{Q}^2$ . It holds:*

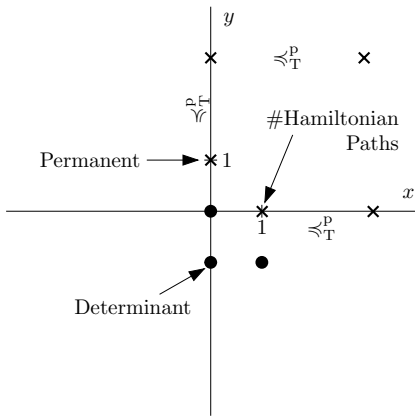
*If  $(a, b) \notin \{(0, 0), (0, -1), (1, -1)\}$ , then  $C(a, b)$  is #P-hard.  
 Otherwise,  $C(a, b)$  is computable in polynomial-time.*

*Proof (outline).* The proof is in several steps (cf. Fig. 1). We begin by classifying the polynomial-time computable points in Section 4. Furthermore, we point out that  $C(0, 1)$  is the permanent and  $C(1, 0)$  is the number of Hamiltonian paths, which both are #P-complete counting problems.

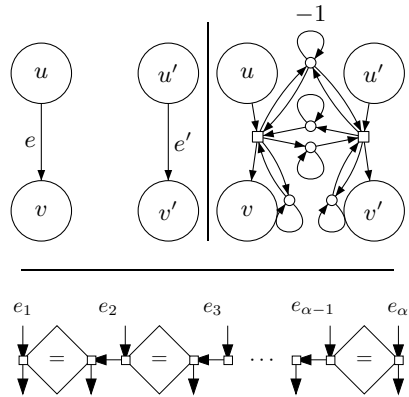
In Section 5, using elementary identities of the cover polynomial and interpolation, we reduce along horizontal lines, that means we prove  $C(0, b) \preceq_T^P C(a, b)$  for all  $a, b$  and  $C(1, 0) \preceq_T^P C(a, 0)$  for all  $a \neq 0$ . This implies the hardness of  $C(a, 1)$  for all  $a$  and of  $C(a, 0)$  for  $a \neq 0$ .

To prove the remaining hardness part where  $b \notin \{-1, 0, 1\}$ , we reduce the permanent to  $C(0, b)$ . Section 6 is the core part of our proof, establishing this reduction  $C(0, 1) \preceq_T^P C(0, b)$  along the  $y$ -axis. There we introduce and analyze the equality gadget, use it to establish a new identity for the *weighted* cover polynomial, and show how to derive a reduction for the standard cover polynomial from this. □

In Section 7, we briefly show how to carry over our result to the geometric version of the cover polynomial.



**Fig. 1.** The big picture: three points (black discs) are easy to evaluate, and the rest of the plane is #P-hard. The crosses indicate the points for the reductions.



**Fig. 2.** Top: Shows how the equality gadget connects two edges  $e, e'$ . Bottom: Shows how the multi-equality gadget connects  $\alpha$  edges with equality gadgets from the top (large diamonds).

### 4 Special Points

A *Hamiltonian path* is a path-cycle cover with exactly one path and zero cycles, and a *cycle cover* is a path-cycle cover without paths. The *permanent*  $\text{Perm}(D)$  is the permanent of the adjacency matrix of  $D$ , and it counts the number of cycle covers of  $D$ . The *determinant*  $\det(D)$  is the determinant  $\det(A)$ . Remarkably, both the determinant and the permanent can be found in the cover polynomial.

**Lemma 1.** *For any nonempty directed graph  $D$ , we have*

- (i)  $C(D, 0, 0) = 0$ ,
- (ii)  $C(D, 1, 0) = \#\text{HAMILTONIANPATHS}(D)$ ,
- (iii)  $C(D, 0, 1) = \text{Perm}(D)$ ,
- (iv)  $C(D, 0, -1) = (-1)^n \det(D)$ ,
- (v)  $C(D, 1, -1) = C(D, 0, -1) - C(D', 0, -1)$  where  $D'$  is derived from  $D$  by adding a new vertex  $v_0$  with all edges to and from the nodes of  $D$ ,

*Proof (sketch).* (For a detailed proof, cf. [11])

The proof of the first three claims is simple. The proof of the fourth claim uses Laplace expansion and the multilinearity of the determinant to show that the contraction-deletion identity from (1)–(3) in [1] is satisfied by the function  $D \mapsto (-1)^{n(D)} \det(D)$ . Since the contraction-deletion identities are defining equations of the cover polynomial, the claim follows.

For the last claim, notice that  $C(D, 1, -1)$  counts all path-cycle covers with at most one path (weighted with  $(-1)^{j(C)}$ ), while the determinant  $C(D, 0, -1)$  counts only cycle covers. The idea is that  $C(D, 1, -1) - C(D, 0, -1)$  is the number of covers of  $D$  with exactly one path, and can be expressed by  $C(D', 0, -1)$ , the number of cycle covers of  $D'$ . This is because every path-cycle cover of  $D$  with one path becomes a cycle cover in  $D'$  where the path gets closed by the  $v_0$ -edges to form a cycle. □

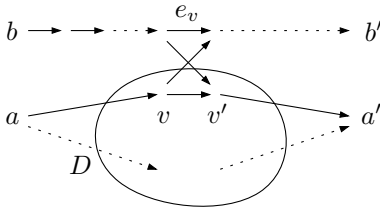
As a consequence,  $C(0, 1)$  and  $C(1, 0)$  are #P-hard [9,12], while  $C(0, 0)$ ,  $C(0, -1)$  and  $C(1, -1)$  are polynomial-time computable. The #P-hardness of  $C(2, -1)$  follows from the following lemma and requires a sophisticated proof.

**Lemma 2.** *It holds  $\#\text{HAMILTONIANPATHS} \preceq_{\text{m}}^{\text{P}} C(2, -1)$ .*

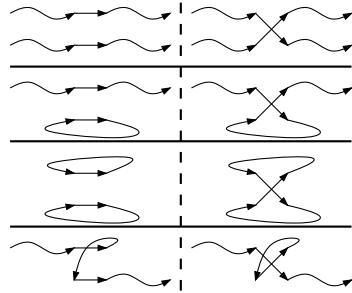
*Proof.* Let  $D$  be a directed graph with vertex set  $V = \{1, \dots, n\}$ . We construct a graph  $D'$  from  $D$  as follows (cf. Fig. 3).

- uncontract all vertices  $v$  from  $D$ , into two vertices  $v, v'$  with one edge  $(v, v')$  in between, and move all outgoing edges  $(v, w)$  to  $(v', w)$ ,
- add fresh vertices  $a, a'$  to  $D$ , and add the edges  $(a, v)$  and  $(v', a')$  for all  $v$ ,
- add an independent directed path of length  $n$  with edges  $e_1, \dots, e_n$ , and
- add the *crossing edges*  $(v, w), (u, v')$  for all  $(v, v')$  and  $e_v = (u, w)$ .

Note that uncontracting edges as above does not change the structure of the path-cycle covers. Therefore, we refer to the graph induced by the vertices  $v$



**Fig. 3.** Shows the graph  $D'$  constructed from  $D$ . The two edges  $e_v, (v, v')$  together with the corresponding crossing edges form the *crossing gadget*.



**Fig. 4.** Shows the change in the number of paths and cycles in a path-cycle cover if we switch one crossing gadget: It is 0 if the two edges are used in distinct paths, and it is 1 otherwise

and  $v'$  again by  $D$ . Let  $b$  be the start vertex of  $e_1$ , and let  $b'$  be the end vertex of  $e_n$ . Since  $a$  and  $b$  have no incoming edges,  $a$  and  $b$  are starting points of paths, and similarly  $a'$  and  $b'$  are ending points of paths in every path-cycle cover.

For the cover polynomial of  $D'$ , we have

$$C(D', 2, -1) = 2 \cdot \sum_C (-1)^{j(C)},$$

where the sum is only over those path-cycle covers of  $D'$  that have exactly two paths and an arbitrary number of cycles.

In the following, we prove  $C(D', 2, -1) = 2^{n+1} \cdot \#\text{HAMILTONIANPATHS}(D)$ .

From a given path-cycle cover  $C$ , related path-cycle covers can be constructed by *switching* the presence of the edges  $e_v, (v, v')$  and their corresponding crossing edges. Let  $\mathcal{C}_0$  be the set of path-cycle covers  $C_0$  of  $D'$  that have exactly two paths, use no crossing edge, and have at least one cycle. We define the set of *bad* cycle covers  $\mathcal{C}_b$  as the closure of  $\mathcal{C}_0$  under switching arbitrary crossing gadgets.

Let  $C \in \mathcal{C}_b$  be arbitrary. By switching, we can uniquely turn  $C$  into a path-cycle cover  $C_0 \in \mathcal{C}_0$ . Let  $v$  be the smallest vertex of  $D$  that occurs in a cycle of  $C_0$ . We define the partner of  $C$  as the path-cycle cover  $p(C) \in \mathcal{C}_b$  which is derived from  $C$  by switching the crossing gadget  $e_v, (v, v')$ . As depicted in Fig. 4, the numbers of cycles in  $C$  and in  $p(C)$  differ by exactly 1.

Since  $p$  is a permutation on  $\mathcal{C}_b$  with the property  $(-1)^{j(C)} = -(-1)^{j(p(C))}$ , the weights  $(-1)^{j(C)}$  of the bad cycle covers sum up to 0 in  $C(D', 2, -1)$ . This means that  $C(D', 2, -1)$  is just 2 times the number of path covers of  $D'$  with exactly two paths. Any such 2-path cover  $C$  of  $D'$  translates to an Hamiltonian path of  $D$  (by switching all gadgets to  $C_0$ , recontracting the edges  $(v, v')$ , and removing  $a, a'$  and the  $b$ - $b'$ -path), and this procedure does not add any cycles. Since there are  $2^n$  possible gadget states, we get

$$C(D', 2, -1) = 2 \cdot 2^n \cdot \#\text{HAMILTONIANPATHS}(D). \quad \square$$

Note that in the proof above, we basically examined an operation necessary for Gaussian elimination: Exchanging two rows or columns in the adjacency matrix switches the sign of the determinant, but as soon as we allow more than one path, this is no longer true for the cover polynomial.

### 5 Horizontal Reductions

Let us consider reductions along the horizontal lines  $L_b := \{(a, b) : a \in \mathbb{Q}\}$ . For a directed graph  $D$ , let  $D^{(r)}$  be the graph obtained by adding  $r$  independent vertices. Corollary 4 in [11] is the core part of the horizontal-line reductions:

$$C(D^{(r)}, x, y) = x^r C(D, x - r, y). \tag{2}$$

From this equation, it is not hard to prove the next lemma (also cf. [11]).

**Lemma 3.** *For all  $(a, b) \in \mathbb{Q}^2$ , we have  $C(0, b) \preceq_T^P C(a, b)$ .*

*Proof (sketch).* For  $a \in \mathbb{N}$ , it follows directly. For  $a \notin \mathbb{N}$ , we can compute the values  $C(D, a - 1, b), \dots, C(D, a - m, b)$  and interpolate to get  $C(D, x, b)$ .  $\square$

In a similar fashion, one can also prove  $C(1, 0) \preceq_T^P C(a, 0)$  for  $a \neq 0$  and  $C(2, -1) \preceq_T^P C(a, 0)$  for  $a \neq 0, 1$ . Please note that  $C(a, b)$  is now known to be #P-hard for every point  $(a, b)$  on the lines  $L_1, L_0$ , and  $L_{-1}$ , except for the three easy points  $(0, 0), (0, -1)$ , and  $(1, -1)$ .

### 6 Vertical Reduction

In this section, we reduce the permanent along the  $y$ -axis:

**Theorem 1.** *Let  $b \in \mathbb{Q}$  with  $-1 \neq b \neq 0$ . It holds  $C(0, 1) \preceq_T^P C(0, b)$ .*

*Proof (outline).* For some graph  $D$ , we compute  $C(D, 0, 1)$  with oracle access to  $C(0, b)$ , and we use interpolation to do so.

In order to interpolate the polynomial  $C(D, 0, y)$ , we need to compute some values  $C(D, 0, b_1), \dots, C(D, 0, b_m)$ . This can be done by using the oracle for some values  $C(D_1, 0, b), \dots, C(D_m, 0, b)$  instead. More specifically, we construct graphs  $D^\alpha$  containing  $\alpha$  copies of a graph  $D$ , such that there is a polynomial-time computable relation between  $C(D, 0, b^\alpha)$  and  $C(D^\alpha, 0, b)$ . Computing  $C(D, 0, b^\alpha)$  for  $\alpha = 1, \dots, m$  and applying interpolation, we get  $C(D, 0, y)$ .

Construction details are spelled out in the remainder of this section.  $\square$

The constructed graph  $D^\alpha$  is a graph in which every cycle cover ideally has  $\alpha$  times the number of cycles a corresponding cycle cover of  $D$  would have. This way, the terms  $y^j$  in the cover polynomial ideally become  $(y^\alpha)^j$ , and some easily computable relation between  $C(D, 0, b^\alpha)$  and  $C(D^\alpha, 0, b)$  can be established.

In the construction, we therefore duplicate the graph  $\alpha$  times, and we connect the duplicates by equality gadgets. These equality gadgets make sure that every cycle cover of  $D^\alpha$  is a cycle cover of  $D$  copied  $\alpha$  times, and thus has roughly  $\alpha$  times the number of cycles. Let us construct the graph  $D^\alpha$  explicitly.



- Start with the input graph  $D$ .
- Create  $\alpha$  copies  $D_1, \dots, D_\alpha$  of  $D$ .
- Let  $e_i$  be the copy of  $e$  in the graph  $D_i$ . Replace every tuple of edges  $(e_1, \dots, e_\alpha)$  by the multiequality gadget on  $\alpha$  edges, which is drawn in Fig. 2.

The  $D_1$ -part of every cycle cover of  $D^\alpha$  is isomorphic to a cycle cover of  $D$  and has to be imitated by the other subgraphs  $D_i$  because of the equality gadgets. What we are left with is to prove that the equality gadget indeed ensures that, in every cycle cover, two edges are either both present or both absent.

Note that some edges in the multiequality gadget become paths of length three. This is because we add the equality gadget first to, say,  $e_1$  and  $e_2$  which gives two new edges each. Next we apply it to one of the new edges for  $e_2$  and to  $e_3$  which explains why  $e_2$  now has three edges. Please also note that the equality gadget can be adapted easily to work as an XOR-gadget [11], easier to analyze than that found in [9].

### 6.1 The Weighted Cover Polynomial

Unfortunately, the equality gadget cannot [1] enforce equality in every possible cycle cover, and we call those cycle covers *good* that satisfy equality (we later change this notion slightly). As you noticed, we introduce weights  $w_e \in \{-1, 1\}$  on the edges. These weights make sure that, in the *weighted* cover polynomial

$$C^w(D, 0, y) := \sum_{\text{cycle cover } C} w(C)y^{j(C)} := \sum_{\text{cycle cover } C} y^{j(C)} \prod_{e \in C} w_e,$$

the *bad* cycle covers sum up to 0, so *effectively* only the good cycle covers are visible. We define the evaluation function  $C^w(a, b)$  in the weighted case again as  $D \mapsto C^w(D, a, b)$  and show that both evaluation complexities are equal.

**Lemma 4.** *For all  $b \in \mathbb{Q}$ , it holds  $C(0, b) \preceq_m^P C^w(0, b)$  and  $C^w(0, b) \preceq_m^P C(0, b)$ .*

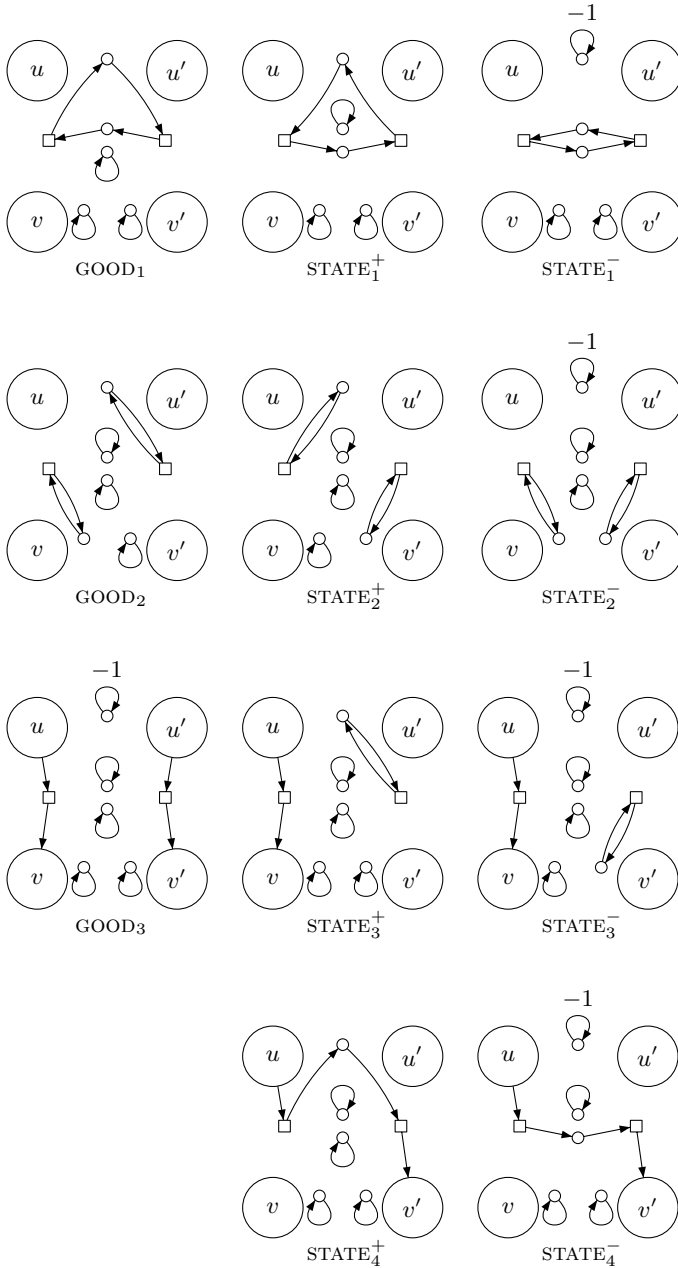
*Proof (sketch).* The first claim is trivial. For the second claim we adapt the proof of Valiant [9] to the cover polynomial: We replace the  $-1$ -edges by an unweighted  $N$ -gadget that simulates a sufficiently large weight  $N$ , and we compute modulo  $N + 1$ . However, our  $N$ -gadget is different from Valiant’s since it has to contribute a constant amount of cycles to cycle covers. To handle rational values, we multiply with the common denominator, and for negative values, we choose  $N$  even larger. (for details, cf. [11]) □

### 6.2 Partner Elimination

In Fig. 5, we have drawn all possible ways for an equality gadget to be covered by cycles if the surrounding graph allows it. The *state* of the gadget is the set

---

<sup>1</sup> *Fortunately!* If it were possible without weights, we could adapt the gadget for a parsimonious reduction from #SAT to the permanent and thereby prove P = NP.



**Fig. 5.** All possible states (=partial cycle covers) of the equality gadget except that the states  $\text{STATE}_3^\pm$  and  $\text{STATE}_4^\pm$  also have symmetric cases, which we have not drawn. We call  $\text{GOOD}_1$ ,  $\text{GOOD}_2$ , and  $\text{GOOD}_3$  *good* states as they have no partners and satisfy the equality property. Note that the choice of the good states is arbitrary as long as the equality property is satisfied and all  $\text{STATE}_i^\pm$  have partners.

of edges chosen to cover it. Notice that the GOOD states as well as STATE<sub>1</sub><sup>±</sup> and STATE<sub>2</sub><sup>±</sup> have the equality property, that is, the left path is completely present or absent if and only if the same holds for the right one.

As stated above, the problem is that the equality gadget does *not* prevent states that do not have the equality property. Therefore, using Lemma 4, we switch to the weighted cover polynomial for which our equality gadget is constructed in such a way, that the good cycle covers contribute a weight  $\neq 0$  to the sum  $C^w(D^\alpha, 0, b)$  while the bad cycle covers do not contribute to the sum, that is, the weights of the bad cycle covers sum up to zero. This is achieved by the fact that every bad cycle cover of weight +1 has a unique bad cycle cover of weight -1 as a partner. More specifically, every cycle cover  $C$  has a partner  $C'$  such that the corresponding summands in the weighted cover polynomial vanish, that is,

$$w(C)y^{j(C)} + w(C')y^{j(C')} = 0.$$

Note that not only the weights must be of different sign, but also the numbers of cycles must be equal! This is the crucial factor why we cannot simply adapt the XOR-gadget of Valiant to form an equality gadget for the cover polynomial. The number of cycles contributed by his XOR-gadget varies a lot, and thus the summands corresponding to the bad cycle covers do not cancel out. (Note that, if we plug in  $y = 1$  to get the permanent, then the condition on the cycles is not needed, and the XOR-gadget works, of course.)

Now let us quickly summarize and prove the properties of the equality gadget. We now call a cycle cover *bad* if an equality gadget is in a state STATE<sub>*i*</sub><sup>±</sup>.

**Lemma 5.** *Every bad cycle cover  $C$  of  $D^\alpha$  has a partner  $C'$  with the properties*

- (i)  $C'$  is again bad, and its partner is  $C$ ,
- (ii) for the weights, it holds  $w(C') = -w(C)$ , and
- (iii) for the number of cycles, it holds  $j(C') = j(C)$ .

*Proof.* We choose an arbitrary ordering on the equality gadgets of  $D^\alpha$ . Let  $C$  be a bad cycle cover and  $g$  be its smallest gadget in state STATE<sub>*i*</sub><sup>±</sup>. We define its partner  $C'$  to be the same cycle cover but with gadget  $g$  in state STATE<sub>*i*</sub><sup>∓</sup> instead. Verifying the three properties proves the claim. □

It immediately follows that only the good cycle covers of  $D^\alpha$  remain in  $C^w(D^\alpha)$ :

$$C^w(D^\alpha, 0, y) = \sum_{\substack{\text{cycle cover } C, \\ C \text{ is good}}} w(C)y^{j(C)}.$$

### 6.3 Counting the Good Cycle Covers

In order to finish the proof of Theorem 1, and thus also the proof of the Main Theorem, it remains to express  $C^w(D^\alpha)$  in an appropriate way in terms of  $C^w(D)$ .

**Lemma 6.** *It holds  $C^w(D^\alpha, 0, y) = (y^{n+4m}(1+y)^{m-n})^{\alpha-1} C^w(D, 0, y^\alpha)$ .*

*Proof (sketch).* Every cycle cover  $C$  of  $D$  induces several possible good cycle covers  $C'$  of  $D^\alpha$ . Since  $|C| = n$ , a number of  $(\alpha - 1)n$  gadgets have state  $\text{GOOD}_3$  in  $C'$ . The other  $m - n$  gadget have the free choice between the states  $\text{GOOD}_1$  and  $\text{GOOD}_2$ . In addition to the  $\alpha j(C)$  large cycles, every gadget contributes small cycles to  $C'$ . More precisely,  $j(C') = \alpha j(C) + 4 \cdot \#\text{GOOD}_1 + 5 \cdot \#\text{GOOD}_2 + 5 \cdot \#\text{GOOD}_3$ . Using the fact that there are  $\binom{m-n}{i}$  possible  $C'$  corresponding to  $C$  and with the property  $\#\text{GOOD}_1(C') = i$ , the claim can be verified by a simple computation (cf. [11]).  $\square$

The easily computable relation between  $C^w(D^\alpha, 0, y)$  and  $C^w(D, 0, y^\alpha)$  from the last lemma proves (a weighted formulation of) Theorem 1.  $\square$

Finally, the Main Theorem follows, for  $-1 \neq b \neq 0$ , from the reduction chain

$$C(0, 1) \preceq_T^P C^w(0, 1) \preceq_T^P C^w(0, b) \preceq_T^P C(0, b) \preceq_T^P C(a, b).$$

## 7 The Geometric Cover Polynomial

The geometric cover polynomial  $C_{\text{geom}}(D, x, y)$  introduced by D’Antona and Munarini [10] is the geometric version of the cover polynomial, that is, the falling factorial  $x^{\underline{2}}$  is replaced by the usual power  $x^i$  in [11]. We are able to give complexity results also for this graph polynomial, except for the line  $L_{-1}$ .

**Theorem 2.** *Let  $(a, b) \in \mathbb{Q}^2$ . It holds:*

*If  $(a, b) \notin \{(0, 0), (*, -1)\}$ , then  $C_{\text{geom}}(a, b)$  is #P-hard.*

*If  $(a, b) \in \{(0, 0), (0, -1)\}$ , then  $C_{\text{geom}}(a, b)$  is computable in polynomial time.*

*Proof (sketch).* The results on the  $y$ -axis follow from  $C(D, 0, y) = C_{\text{geom}}(D, 0, y)$ . For the other points, a new horizontal reduction must be established. This can be done by considering the  $\alpha$ -fattening of the graph, in which every edge is replaced by  $\alpha$  parallel edges (for details, cf. [11]).  $\square$

## 8 Conclusion and Further Work

In this paper, we completely characterized the complexity of evaluating the cover polynomial in the rational plane. Our reductions should also work for complex or algebraic numbers, but we do not have any interpretation for such points yet.

The next natural step along the path is to characterize the complexity of approximately evaluating the cover polynomial, in analogy to recent results by Goldberg and Jerrum [6] for the Tutte polynomial. Very recently, Fouz had success in characterizing the approximability of the geometric cover polynomial in large regions of the plane.

One might conjecture that the connection between the Tutte and the cover polynomial is solved because both are #P-hard to evaluate at most points, so they can be computed from each other (where they are in #P). From a combinatorial point of view, however, #P-reductions are not really satisfying steps

towards a deeper understanding of the connection between the two graph polynomials.

One promising approach might be to consider generalizations of the cover polynomial. Very recently, Courcelle analyzed general contraction-deletion identities analogous to those of the coloured Tutte polynomial, and he found conditions for the variables which are necessary and sufficient for the well-definedness of the polynomial. Unfortunately, these conditions seem too restrictive.

The contraction-deletion identities of Tutte and cover polynomial look astonishingly similar, but contractions work differently for directed and for undirected graphs. Explaining the connection between Tutte and cover polynomial means overcoming this difference.

**Acknowledgements.** We would like to thank Mahmoud Fouz, Thomas Jansen and Moritz Hardt for fruitful discussions and comments on earlier versions of this paper.

## References

1. Chung, F.R., Graham, R.L.: On the cover polynomial of a digraph. *Journal of Combinatorial Theory Series B* 65, 273–290 (1995)
2. Jaeger, F., Vertigan, D.L., Welsh, D.J.: On the computational complexity of the Jones and Tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society* 108, 35–53 (1990)
3. Lotz, M., Makowsky, J.A.: On the algebraic complexity of some families of coloured Tutte polynomials. *Advances in Applied Mathematics* 32, 327–349 (2004)
4. Bollobás, B., Riordan, O.: A Tutte polynomial for coloured graphs. *Combinatorics, Probability and Computing* 8, 45–93 (1999)
5. Giménez, O., Noy, M.: On the complexity of computing the Tutte polynomial of bicircular matroids. *Combinatorics, Probability and Computing* 15, 385–395 (2006)
6. Goldberg, L.A., Jerrum, M.: Inapproximability of the Tutte polynomial (2006)
7. Bulatov, A., Grohe, M.: The complexity of partition functions. *Theoretical Computer Science* 348, 148–186 (2005)
8. Dyer, M.E., Goldberg, L.A., Paterson, M.: On counting homomorphisms to directed acyclic graphs. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) *ICALP 2006*. LNCS, vol. 4051, pp. 38–49. Springer, Heidelberg (2006)
9. Valiant, L.G.: The complexity of computing the permanent. *Theoretical Computer Science* 8, 189–201 (1979)
10. D’Antona, O.M., Munarini, E.: The cycle-path indicator polynomial of a digraph. *Adv. Appl. Math.* 25, 41–56 (2000)
11. Bläser, M., Dell, H.: Complexity of the cover polynomial (journal version) (to appear)
12. Dyer, M.E., Frieze, A.M., Jerrum, M.: Approximately counting Hamilton paths and cycles in dense graphs. *SIAM Journal on Computing* 27, 1262–1272 (1998)

# A Generalization of Cobham's Theorem to Automata over Real Numbers

Bernard Boigelot and Julien Brusten\*

Institut Montefiore, B28  
Université de Liège  
B-4000 Liège, Belgium  
{boigelot,brusten}@montefiore.ulg.ac.be

**Abstract.** This paper studies the expressive power of finite-state automata recognizing sets of real numbers encoded positionally. It is known that the sets that are definable in the first-order additive theory of real and integer variables  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$  can all be recognized by weak deterministic Büchi automata, regardless of the encoding base  $r > 1$ . In this paper, we prove the reciprocal property, i.e., that a subset of  $\mathbb{R}$  that is recognizable by weak deterministic automata in every base  $r > 1$  is necessarily definable in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ . This result generalizes to real numbers the well-known Cobham's theorem on the finite-state recognizability of sets of integers. Our proof gives interesting insight into the internal structure of automata recognizing sets of real numbers, which may lead to efficient data structures for handling these sets.

## 1 Introduction

The verification of infinite-state systems, in particular the reachability analysis of systems modeled as finite-state machines extended with unbounded variables, has prompted the development of symbolic data structures for representing the sets of values that have to be handled during state-space exploration [Boi98].

A simple representation strategy consists in using finite-state automata: The values in the considered domain are encoded as words over a given finite alphabet; a set of values is thus encoded as a language. If this language is regular, then a finite-state automaton that accepts it forms a representation of the set [WB98].

This approach has many advantages: Regular languages are closed under all usual set-theory operators (intersection, union, complement, Cartesian product, projection, ...), and automata are easy to manipulate algorithmically. Deterministic automata can also be reduced to a canonical form, which simplifies comparison operations between sets.

The expressive power of automata is also well suited for verification applications. In the case of programs manipulating unbounded integer variables, it is known for a long time that the sets of integers that can be recognized by

---

\* Research fellow ("aspirant") of the Belgian Fund for Scientific Research (F.R.S.-FNRS).

a finite-state automaton using the positional encoding of numbers in a base  $r > 1$  correspond to those definable in an extension of Presburger arithmetic, i.e., the first-order additive theory of the integers  $\langle \mathbb{Z}, +, < \rangle$  [Büc62]. Furthermore, the well known Cobham's theorem characterizes the sets that are representable by automata in all bases  $r > 1$  as being exactly those that are Presburger-definable [Cob69, BHMV94].

In order to analyze systems relying on integer and real variables, such as timed or hybrid automata, automata-based representations of numbers can be generalized to real values [BBR97]. From a theoretical point of view, this amounts to moving from finite-word to infinite-word automata, which is not problematic. It has been shown that the sets of reals that can be recognized by infinite-word automata in a given encoding base are those definable in an extension of the first-order additive theory of real and integers variables  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$  [BRW98].

In practice though, handling infinite-word automata can be difficult, especially if set complementation needs to be performed. It is however known that, for representing the sets definable in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ , the full expressive power of Büchi automata is not required, and that the much simpler subclass of *weak deterministic* automata is sufficient [BJW05]. The advantage is that, from an algorithmic perspective, handling weak automata is similar to manipulating finite-word automata.

A natural question is then to characterize precisely the expressive power of weak deterministic automata representing sets of real numbers. For a given encoding base  $r > 1$ , it is known that the representable sets form a base-dependent extension of  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ . This covers, in particular, all the sets definable in  $\langle \mathbb{R}, \mathbb{Z}, +, <, P_r \rangle$ , where  $P_r$  is a predicate that checks whether its argument is a power of  $r$  [Bru06].

This paper is aimed at characterizing the subsets of  $\mathbb{R}$  that can be represented as weak deterministic automata in multiple bases. Our central result is to show that, for two relatively prime bases  $r_1$  and  $r_2$ , the sets that are simultaneously recognizable in bases  $r_1$  and  $r_2$  can be defined in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ . As a corollary, such sets are then representable in any base  $r > 1$ .

The intuition behind our proof is the following. First, we reduce the problem to characterizing the representable subsets of  $[0, 1]$ . We then introduce the notion of interval boundary points, as points with special topological properties, and establish that a set representable in multiple bases can only contain finitely many such points. Finally, we show that this property implies that  $S$  is definable in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ . The argument used for this last step provides a description of the internal structure of automata representing sets definable in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ . This result may help to develop efficient data structures for handling such sets.

## 2 Representing Sets of Numbers with Automata

In this section, we briefly present the automata-based representations of sets of integer and real values.

### 2.1 Number Decision Diagrams

Let  $r > 1$  be an integer *base*. A natural number  $x \in \mathbb{N}$  can be *encoded* positionally in base  $r$  by finite words  $b_{p-1}b_{p-2} \dots b_1b_0$  over the alphabet  $\Sigma_r = \{0, 1, \dots, r-1\}$ , such that  $x = \sum_{i=0}^{p-1} b_i r^i$ . Negative values are encoded by their  $r$ ’s-complement, i.e., the encodings of  $x \in \mathbb{Z}$  with  $x < 0$  are formed by the last  $p$  digits of the encodings of  $r^p + x$ . The length  $p$  of the encodings of a number  $x \in \mathbb{Z}$  is not fixed, but must be non-zero and large enough for  $-r^{p-1} \leq x < r^{p-1}$  to hold. As a consequence, the most significant digit of encodings, called the *sign digit*, is equal to  $r - 1$  for strictly negative numbers, and to 0 for positive numbers.

This encoding scheme maps a subset  $S$  of  $\mathbb{Z}$  onto a language over  $\Sigma_r$ . If the language containing all the encodings of the elements of  $S$  is regular, then a finite-state automaton that accepts it is called a *Number Decision Diagram (NDD)*, and is said to represent, or recognize, the set  $S$ . NDDs can be generalized to representing subsets of  $\mathbb{Z}^n$ , i.e., sets of vectors, for any  $n > 0$  [Büc62, WB95, Boi98].

It has been shown [Büc62, Vil92, BHMV94] that the subsets of  $\mathbb{Z}$  recognizable by NDDs in a base  $r > 1$  are exactly those that can be defined in the first-order theory  $\langle \mathbb{Z}, +, <, V_r \rangle$  where  $V_r(x)$  is the function mapping an integer  $x > 0$  to the greatest power of  $r$  dividing it. Moreover, the sets that are recognizable by NDDs in every base  $r > 1$  have been characterized by Cobham [Cob69] as being exactly those that are definable in  $\langle \mathbb{Z}, +, < \rangle$ , i.e., Presburger arithmetic. This result has been extended to subsets of  $\mathbb{Z}^n$  by Semenov [Sem77].

Computing the intersection, union, complementation, difference and Cartesian product of sets represented by NDDs reduces to performing the corresponding operations on the languages accepted by the automata. Projection is more tricky, as the resulting automaton has to be completed in order to accept all the encodings of the vectors it recognizes. Finally, since NDDs are finite-word automata, they can be determinized, as well as minimized into a canonical form.

### 2.2 Real Number Automata

Real numbers can also be encoded positionally. Let  $r > 1$  be a base. An encoding  $w$  of a number  $x \in \mathbb{R}$  is an infinite word  $w_I \cdot \star \cdot w_F$  over  $\Sigma_r \cup \{\star\}$ , where  $w_I \in \Sigma_r^*$  encodes the integer part  $x_I \in \mathbb{Z}$  of  $x$ , and  $w_F \in \Sigma_r^\omega$  its fractional part  $x_F \in [0, 1]$ , i.e., we have  $w_F = b_1b_2b_3 \dots$  with  $x_F = \sum_{i>0} b_i r^{-i}$ . Note that some numbers have two distinct encodings with the same integer-part length. For example, in base 10, the number 11/2 has the encodings  $0^+ \cdot 5 \cdot \star \cdot 5 \cdot 0^\omega$  and  $0^+ \cdot 5 \cdot \star \cdot 4 \cdot 9^\omega$ . Such encodings are said to be *dual*. We denote by  $A_r$  the set of valid prefixes of base- $r$  encodings that include a separator, i.e.,  $A_r = \{0, r - 1\} \cdot \Sigma_r^* \cdot \star \cdot \Sigma_r^*$ .

Similarly to the case of integers, the base- $r$  encoding scheme transforms a set  $S \subseteq \mathbb{R}$  into a language  $L(S) \subseteq A_r \cdot \Sigma_r^\omega$ . A *Real Number Automaton (RNA)* is defined as a Büchi automaton that accepts the language containing all the base- $r$  encodings of the elements of  $S$ . This representation can be generalized into *Real Vector Automata (RVA)*, suited for subsets of  $\mathbb{R}^n$  ( $n > 0$ ) [BBR97].

The expressiveness of RVA (and RNA) has been studied [BRW98]: The subsets of  $\mathbb{R}^n$  that are representable in a base  $r > 1$  are exactly those that are definable



in the first-order theory  $\langle \mathbb{R}, \mathbb{Z}, +, <, X_r \rangle$ , where  $X_r(x, u, k)$  is a base-dependent predicate that is true iff  $u$  is an integer power of  $r$ , and there exists an encoding of  $x$  in which the digit at the position specified by  $u$  is equal to  $k$ . The predicate  $X_r$  can alternatively be replaced by a function  $V_r$  analogous to the one defined in the integer case [Bru06]: We say that  $x \in \mathbb{R}$  divides  $y \in \mathbb{R}$  iff there exists an integer  $k$  such that  $kx = y$ . The function  $V_r$  is then defined such that  $V_r(x)$  returns the greatest power of  $r$  dividing  $x$ , if it exists, and 1 otherwise.

### 2.3 Weak Deterministic RNA

As in the case of integers, applying most set-theory operators to RNA (or RVA) reduces to carrying out the same operations on their accepted language. This is somehow problematic, since operations like set complementation are typically costly and tricky to implement on infinite-word automata [KV05].

In order to alleviate this problem, it has been shown that the full expressive power of Büchi automata is not needed for representing the subsets of  $\mathbb{R}^n$  ( $n \geq 0$ ), that are definable in the first-order additive theory  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$  of mixed integer and real variables [BJW05]. Such sets can indeed be represented by *weak deterministic* RVA, i.e., deterministic RVA such that their set of states can be partitioned into disjoint subsets  $Q_1, \dots, Q_m$ , where each  $Q_i$  contains only either accepting or non-accepting states, and there exists a partial order  $\leq$  on the sets  $Q_1, \dots, Q_m$  such that for every transition  $(q, a, q')$  of the automaton, with  $q \in Q_i$  and  $q' \in Q_j$ , we have  $Q_j \leq Q_i$ .

As remarked in [W193], weak deterministic automata are infinite-word automata that can be manipulated essentially in the same way as finite-word ones. There exist efficient algorithms for applying to weak deterministic RVA all classical set-theory operators (intersection, union, complement, Cartesian product, projection, ...) [BJW05]. Furthermore, such RVA can be minimized into a canonical form.

It is worth mentioning that expressiveness of weak deterministic RVA is clearly not limited to the sets that are definable in the first-order additive theory of the integers and reals. For instance, the set of (negative and positive) integer powers of the representation base is clearly recognizable. Let  $r > 1$  be a base, and  $P_r(x)$  be a predicate that holds iff  $x$  is an integer power of  $r$ . It has been shown, using a quantifier elimination result for  $\langle \mathbb{R}, 1, +, \leq, P_r \rangle$  [vdD85, AY07], that all the sets definable in  $\langle \mathbb{R}, \mathbb{Z}, +, <, P_r \rangle$  can also be represented by weak deterministic RVA in base  $r$  [Bru06].

## 3 Problem Reduction

Let  $S \subseteq \mathbb{R}$  be a set recognizable by a weak deterministic RNA  $\mathcal{A}_1$ , assumed to be in canonical form, in a base  $r > 1$ . Each accepting path of  $\mathcal{A}$  contains exactly one occurrence of the separator symbol  $\star$ . Each transition labeled by  $\star$  thus links two distinct strongly connected components of  $\mathcal{A}$ . Since there are only finitely many such transitions, the language  $L$  accepted by  $\mathcal{A}$  is of the form  $\bigcup_i L_i^I \cdot \star \cdot L_i^F$ ,

where the union is finite, and for all  $i$ ,  $L_i^I \subseteq \Sigma_r^*$  encodes the integer part, and  $L_i^F \subseteq \Sigma_r^\omega$  the fractional part, of the encodings of numbers  $x \in S$ . More precisely, for every  $i$ , let  $S_i^I \subseteq \mathbb{Z}$  denote the set encoded by  $L_i^I$  and let  $S_i^F \subseteq [0, 1]$  denote the set encoded by  $0^+ \cdot \star \cdot L_i^F$ . We have  $S = \bigcup_i (S_i^I + S_i^F)$ . Note that each  $L_i^I$  is recognizable by a NDD in base  $r$  and that, similarly, each language of the form  $0^+ \cdot \star \cdot L_i^F$  is recognizable by a RNA (except for the dual encodings of 0 and 1, which can be explicitly added to the language if needed).

The decomposition of  $S$  into sets  $S_i^I$  and  $S_i^F$  of integer and fractional parts does not depend on the representation base. Therefore, if  $S$  is recognizable in two relatively prime bases  $r_1$  and  $r_2$ , then so are  $S_i^I$  and  $S_i^F$  for every  $i$ . From Cobham’s theorem, each  $S_i^I$  must then be definable in  $\langle \mathbb{Z}, +, < \rangle$ . In order to show that  $S$  is definable in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ , it is hence sufficient to prove that each  $S_i^F$  is definable in that theory. We have thus reduced the problem of characterizing the subsets of  $\mathbb{R}$  that are simultaneously recognizable in two relatively prime bases to the same problem over the subsets of  $[0, 1]$ .

### 4 Interval Boundary Points

We now consider a set  $S \subseteq [0, 1]$  represented by a weak deterministic RNA  $\mathcal{A}$ . We define the *interval boundary points* of  $S$  as points with specific topological properties, and establish a relation between the existence of such points and some structures in the transition graph of  $\mathcal{A}$ .

#### 4.1 Definitions

A *neighborhood*  $N_\varepsilon(x)$  of a point  $x \in \mathbb{R}$ , with  $\varepsilon > 0$ , is the set  $N_\varepsilon(x) = \{y \mid |x - y| < \varepsilon\}$ . A point  $x \in \mathbb{R}$  is a *boundary point* of  $S$  iff all its neighborhoods contain points from  $S$  as well as from its complement  $\overline{S}$ , i.e.,  $\forall \varepsilon > 0 : N_\varepsilon(x) \cap S \neq \emptyset \wedge N_\varepsilon(x) \cap \overline{S} \neq \emptyset$ .

A *left neighborhood*  $N_\varepsilon^<(x)$  of a point  $x \in \mathbb{R}$ , with  $\varepsilon > 0$ , is the set  $N_\varepsilon^<(x) = \{y \mid x - \varepsilon < y < x\}$ . Similarly, a *right neighborhood*  $N_\varepsilon^>(x)$  of  $x$  is defined as  $N_\varepsilon^>(x) = \{y \mid x < y < x + \varepsilon\}$ . A boundary point  $x$  of  $S$  is a *left interval boundary point* of  $S$  iff it admits a left neighborhood  $N_\varepsilon^<(x)$  that is entirely contained in either  $S$  or  $\overline{S}$ , i.e.,  $\exists \varepsilon > 0 : N_\varepsilon^<(x) \subseteq S \vee N_\varepsilon^<(x) \subseteq \overline{S}$ . *Right interval boundary points* are defined in the same way. A point  $x \in S$  is an *interval boundary point* of  $S$  iff it is a left or a right interval boundary point of  $S$ .

Each interval boundary point  $x$  of  $S$  is thus characterized by its direction (left or right), its polarity w.r.t.  $S$  (i.e., whether  $x \in S$  or  $x \notin S$ ), and the polarity of its left or right neighborhoods of sufficiently small size (i.e., whether they are subsets of  $S$  or of  $\overline{S}$ ). The possible combinations define eight *types* of interval boundary points, that are illustrated in Figure [11](#).

#### 4.2 Recognizing Interval Boundary Points

Recall that  $\mathcal{A}$  is a weak deterministic RNA recognizing a set  $S \subseteq [0, 1]$ . We assume w.l.o.g. that  $\mathcal{A}$  is canonical and complete, in the sense that from each state

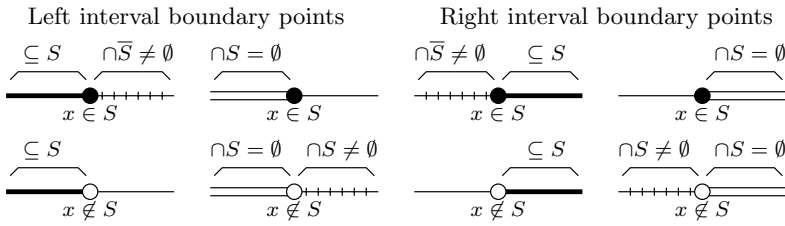


Fig. 1. Types of interval boundary points

$q$  and alphabet symbol  $a$ , there exists an outgoing transition from  $q$  labeled by  $a$ . Consider a path  $\pi$  of  $\mathcal{A}$  that reads an encoding  $w$  of a left interval boundary point  $x$  of  $S$ . Since  $\mathcal{A}$  is weak,  $\pi$  eventually reaches a strongly connected component  $C$  that it does not leave. The accepting status of  $C$  corresponds to the polarity of  $x$  w.r.t.  $S$ .

Since  $x$  is a left interval boundary point, all its sufficiently small left neighborhoods are either subsets of  $S$  or subsets of  $\bar{S}$ , depending on the type of  $x$ . Hence, from each state  $s$  of  $C$  visited infinitely many times by  $\pi$ , its outgoing transitions labeled by smaller digits than the one read in  $\pi$  must necessarily lead to either the universal or the empty strongly connected component of  $\mathcal{A}$ . It follows that, after having reached some state  $s$  in  $C$ , the path  $\pi$  follows the transitions within  $C$  that are labeled by the smallest possible digits, hence it eventually cycles through a loop. A similar result holds for right interval boundary points, which are read by paths that eventually follow the largest possible digits in their terminal strongly connected component.

As a consequence, every base- $r$  encoding  $w$  of an interval boundary point of  $S$  is necessarily *ultimately periodic*, i.e., such that  $w = u \cdot v^\omega$ , with  $u \in A_r$  and  $v \in \Sigma_r^+$ . Besides, each ultimate period  $v$  of such encodings can be uniquely determined from a suitable state of  $\mathcal{A}$  associated with a direction (left or right). We therefore have the following results.

**Theorem 1.** *Each interval boundary point of a subset of  $[0, 1]$  that is recognizable by a weak deterministic RNA is a rational number.*

**Theorem 2.** *Let  $S \subseteq [0, 1]$  be a set recognizable by a weak deterministic RNA in a base  $r > 1$ . The set of ultimate periods of the base- $r$  encodings of the interval boundary points of  $S$  is finite.*

### 4.3 Recognizing Interval Boundary Points in Multiple Bases

Consider now a set  $S \subseteq [0, 1]$  that is simultaneously recognizable by weak deterministic RNA in two relatively prime bases  $r_1 > 1$  and  $r_2 > 1$ . Let  $\mathcal{A}_1$  and  $\mathcal{A}_2$  denote, respectively, such RNA.

Suppose that  $S$  has infinitely many interval boundary points. From Theorem 2, there must exist some ultimate period  $v \in \Sigma_{r_1}^+$  such that infinitely many

interval boundary points of  $S$  have base- $r_1$  encodings of the form  $u_i \cdot v^\omega$ , with  $\forall i : u_i \in A_{r_1}$ . Moreover, the language  $L$  of the words  $u_i$  for which  $u_i \cdot v^\omega$  encodes an interval boundary point of  $S$ , and such that  $u_i$  and  $v$  do not end with the same digit, is infinite and regular. (The restriction on the last digit of  $u_i$  and  $v$  expresses that  $u_i$  is the smallest aperiodic prefix of  $u_i \cdot v^\omega$ .) Indeed, each  $u_i \in L$  can be recognized by a path from the initial state of  $\mathcal{A}$  to a state from which  $v$  can be read as the ultimate period of an encoding of an interval boundary point.

Hence, there exist  $w_1 \in A_{r_1}$  and  $w_2, w_3 \in \Sigma_{r_1}^*$ , with  $|w_2| > 0$ , such that  $\forall k : w_1 \cdot (w_2)^k \cdot w_3 \in L$ . Furthermore, we have that  $w_2 \cdot w_3$  and  $v$  do not end with the same digit.

Thus, for each  $k \geq 0$ , there exists an interval boundary point of  $S$  with a base- $r_1$  encoding of the form  $w_1 \cdot (w_2)^k \cdot w_3 \cdot v^\omega$ . Each word in this language is ultimately periodic, thus it encodes in base  $r_1$  a rational number that can also be encoded by an ultimately periodic word in base  $r_2$ . We use the following lemma.

**Lemma 1.** *Let  $r_1 > 1$  and  $r_2 > 1$  be relatively prime bases, and let  $w_1 \in A_{r_1}, w_2, w_3, w_4 \in \Sigma_{r_1}^*$ , with  $|w_2| > 0, |w_4| > 0$ , such that the words  $w_2 \cdot w_3$  and  $w_4$  do not end with the same digit. The subset of  $\mathbb{Q}$  encoded in base  $r_1$  by the language  $w_1 \cdot (w_2)^* \cdot w_3 \cdot (w_4)^\omega$  cannot be encoded in base  $r_2$  with only a finite number of ultimate periods.*

*Proof.* The proof is given in Appendix [A](#). □

Together with Theorem [2](#), this lemma contradicts our assumption that  $S$  has infinitely many interval boundary points. We thus have the following theorem.

**Theorem 3.** *If a set  $S \subseteq [0, 1]$  is simultaneously recognizable by weak deterministic RNA in two relatively prime bases, then it has finitely many interval boundary points.*

We therefore call a set that satisfies the hypotheses of Theorem [3](#) a *finite-boundary set*.

## 5 Finite-Boundary Sets

Our goal is now to characterize the structure of the transition graph of RNA that recognize finite-boundary sets. We start by establishing some properties that hold for all weak deterministic RNA, and then focus on the specific case of finite-boundary sets.

### 5.1 Properties of Weak Deterministic RNA

Let  $\mathcal{A}$  be a weak deterministic RNA, which we assume to be complete and canonical, recognizing a subset of  $\mathbb{R}$  in a base  $r > 1$ . Consider a strongly connected component  $C$  of  $\mathcal{A}$  such that each of its outgoing transitions leads to either the universal or the empty strongly connected component, i.e., those accepting respectively the languages  $\Sigma_r^\omega$  and  $\emptyset$ .

**Lemma 2.** *Let  $\pi$  be a minimal (resp. maximal) infinite path within  $C$ , i.e., a path that follows from each visited state the transition of  $C$  labeled by the smallest (resp. largest) possible digit. The destination of all outgoing transitions from states visited by  $\pi$ , and that are labeled by a smaller (resp. larger) digit than the one read in  $\pi$ , is identical.*

*Proof.* We first study the case of two transitions  $t_1$  and  $t_2$  originating from the same state  $s$  visited by  $\pi$ , that are respectively labeled by digits  $d_1, d_2$  smaller than the digit  $d$  read from  $s$  in  $\pi$ . Among the digits that satisfy this condition, one can always find consecutive values, hence it is sufficient to consider the case where  $d_2 = d_1 + 1$ .

Let  $\sigma$  be a finite path that reaches  $s$  from the initial state of  $\mathcal{A}$ . By appending to  $\sigma$  suffixes that read  $d_1 \cdot (r-1)^\omega$  and  $d_2 \cdot 0^\omega$ , one obtains paths that recognize dual encodings of the same number, hence these paths must be either both accepting or both non-accepting. Therefore,  $t_1$  and  $t_2$  share the same destination.

Consider now transitions  $t_1$  and  $t_2$  from distinct states  $s_1$  and  $s_2$  visited by  $\pi$ , labeled by smaller digits than those – respectively denoted  $d_1$  and  $d_2$  – read in  $\pi$ . We can assume w.l.o.g. that  $s_1$  and  $s_2$  are consecutive among the states visited by  $\pi$  that have such outgoing transitions. In other words, the subpath of  $\pi$  that links  $s_1$  to  $s_2$  is labeled by a word of the form  $d_1 \cdot 0^k$ , with  $d_1 > 0$  and  $k \geq 0$ .

Let  $\sigma'$  be a finite path that reaches  $s_1$  from the initial state of  $\mathcal{A}$ . Appending to  $\sigma'$  suffixes that read  $(d_1 - 1) \cdot (r - 1)^\omega$  and  $d_1 \cdot 0^\omega$  yields paths that read dual encodings of the same number, hence these paths must be either both accepting or both non-accepting. The destinations of the transitions that leave  $C$  from  $s_1$  and  $s_2$  must thus be identical.

The case of maximal paths is handled in the same way. □

The following result now expresses a constraint on the trivial (acyclic) strongly connected components of the fractional part of  $\mathcal{A}$  (i.e., the part of  $\mathcal{A}$  reached after reading an occurrence of the symbol  $\star$ ).

**Lemma 3.** *From any trivial strongly connected component of the fractional part of  $\mathcal{A}$ , there must exist a reachable strongly connected component that is neither empty, trivial, nor universal.*

*Proof.* The proof is by contradiction. Let  $\{s\}$  be a trivial strongly connected component of the fractional part of  $\mathcal{A}$ . Assume that all paths from  $s$  eventually reach the universal or the empty strongly connected component, after passing only through trivial components. As a consequence, the language accepted from  $s$  is of the form  $L \cdot \Sigma_r^\omega$ , where  $L \subset \Sigma_r^*$  is finite. We can require w.l.o.g. that all words in  $L$  share the same length  $l$ . Note that  $L$  cannot be empty or equal to  $\Sigma_r^l$ , since  $s$  does not belong to the empty or universal components.

Each word in  $\Sigma_r^l$  can be seen as the base- $r$  encoding of an integer in the interval  $[0, r^l - 1]$ . Since  $L$  is neither empty nor universal, there exist two words  $w_1, w_2 \in \Sigma_r^l$  that do not both belong to  $L$  or to  $\Sigma_r^l \setminus L$ , and that encode two consecutive integers  $n$  and  $n + 1$ . Then,  $u \cdot w_2 \cdot 0^\omega$  and  $u \cdot w_1 \cdot (r - 1)^\omega$  encode

the same number in base  $r$ , where  $u$  is the label of an arbitrary path from the initial state of  $\mathcal{A}$  to  $s$ . This contradicts the fact that  $\mathcal{A}$  accepts all the encodings of the numbers it recognizes.  $\square$

### 5.2 Properties of RNA Recognizing Finite-Boundary Sets

**Theorem 4.** *Let  $\mathcal{A}$  be a weak deterministic RNA, supposed to be in complete and canonical form, recognizing a finite-boundary set  $S \subseteq [0, 1]$ . Each non-trivial, non-empty and non-universal strongly connected component of the fractional part of  $\mathcal{A}$  takes the form of a single cycle. Moreover, from each such component, the only reachable strongly connected components besides itself are the empty or the universal ones.*

*Proof.* Let  $C$  be a non-trivial, non-empty and non-universal strongly connected component of the fractional part of  $\mathcal{A}$ , and let  $s$  be an arbitrary state of  $C$ . The path  $\pi$  from  $s$  that stays within  $C$  and follows the transitions with the smallest possible digits is cyclic, and determines the ultimate period of encodings of some interval boundary points of  $S$ . If  $C$  contains other cycles, or if  $C$  is reachable from other non-trivial strongly connected components in the fractional part, then  $\pi$  can be prefixed by infinitely many reachable paths from an entry state of the fractional part of  $\mathcal{A}$  to  $s$ . This contradicts the fact that  $S$  has only finitely many interval boundary points. That no trivial strongly connected component can be reachable from  $C$  then follows from Lemma 3.  $\square$

This result characterizes quite precisely the shape of the fractional part of a weak deterministic RNA recognizing a finite-boundary set: Its transition graph is first composed of a bottom layer of strongly connected components containing only the universal and the empty one, and then a (possibly empty) layer of single-cycle components leading to the bottom layer. Thanks to Lemma 2, the transitions that leave a single-cycle component with a smaller (or larger) digit all lead to the same empty or universal component (which may differ for the smaller and larger cases). Thus, each single-cycle component can simply be characterized by its label and the polarity of its smaller and greater alternatives. Finally, the two layers of non-trivial strongly connected components can be reached through an acyclic structure of trivial components, such that from each of them, there is at least one outgoing path leading to a single-cycle component.

As a consequence, we are now able to describe the language accepted by such a RNA.

**Theorem 5.** *Let  $\mathcal{A}$  be a weak deterministic RNA recognizing a finite-boundary set  $S \subseteq [0, 1]$  encoded in a base  $r > 1$ . The language  $L$  accepted by  $\mathcal{A}$  can be expressed as*

$$L = \bigcup_i L' \cdot w_i \cdot \Sigma_r^\omega \cup \bigcup_i L' \cdot w'_i \cdot (v_i)^\omega \cup \bigcup_i L' \cdot w''_i \cdot (\Sigma_r^\omega \setminus (v_i)^\omega) \cup L_0 \cup L_1,$$

where each union is finite,  $\forall i : w_i, w'_i, w''_i, v_i, v'_i \in \Sigma_r^*$  with  $|v_i| > 0, |v'_i| > 0, L' = 0^+ \cdot \star, L_0$  is either empty or equal to  $(r - 1)^+ \cdot \star \cdot (r - 1)^\omega$ , and  $L_1$  is either empty or equal to  $0^+ \cdot 1 \cdot \star \cdot 0^\omega$ .

(The terms  $L_0$  and  $L_1$  are introduced in order to deal with the dual encodings of 0 and 1.)

In the expression given by Theorem 5, each term of the union encodes a subset of  $[0, 1]$  that is definable in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ :  $L' \cdot w_i \cdot \Sigma_r^\omega$  defines an interval  $[a, b]$ , with  $a, b \in \mathbb{Q}$ , the terms  $L' \cdot w'_i \cdot (v_i)^\omega, L_0$  and  $L_1$  correspond to single rational numbers  $c \in \mathbb{Q}$ , and  $L' \cdot w''_i \cdot (\Sigma_r^\omega \setminus (v'_i)^\omega)$  recognizes a set  $[a, b] \setminus \{c\}$  with  $a, b, c \in \mathbb{Q}$ . This shows that the set  $S \subseteq [0, 1]$  recognized by  $\mathcal{A}$  is definable in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ . Combining this result with Theorem 3, as well as the reduction discussed in Section 3, we get our main result:

**Theorem 6.** *If a set  $S \subseteq \mathbb{R}$  is simultaneously recognizable by weak deterministic RNA in two relatively prime bases, then it is definable in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ .*

**Corollary 1.** *A set  $S \subseteq \mathbb{R}$  is recognizable by weak deterministic RNA in every base  $r > 1$  iff it is definable in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ .*

## 6 Conclusions and Future Work

The main contribution of this work is to show that the subsets of  $\mathbb{R}$  that can be recognized by weak deterministic RNA in all integer bases  $r > 1$  are exactly those that are definable in the first-order additive theory of the real and integer numbers  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ . Our central result is actually stronger, stating that recognizability in two relatively prime bases  $r_1$  and  $r_2$  is sufficient for forcing definability in  $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$ . Using the same argument as in the proof of Lemma 1, this result can directly be extended to bases  $r_1$  and  $r_2$  that do not share the same set of prime factors. This differs slightly from the statement of Cobham’s original theorem, which considers instead bases that are multiplicatively independent, i.e., that cannot be expressed as integer powers of the same integer [Cob69, BHMV94]. Unfortunately, our approach does not easily generalize to multiplicatively independent bases, since Theorem 3 then becomes invalid. Addressing this issue is an interesting open problem.

Another contribution is a detailed characterization of the transition graph of weak deterministic RNA that represent subsets of  $\mathbb{R}$  defined in first-order additive arithmetic. This characterization could be turned into efficient data structures for handling such RNA. In particular, since their fractional parts recognize a finite union of interval and individual rational values, an efficient representation might be based on symbolic data structures such as BDDs for handling large but finite enumerations. Another possible application is the extraction of formulas from automata-based representations of sets [Lat05, Ler05].

Finally, another goal will be to extend our results to sets in higher dimensions, i.e., to generalize Semenov’s theorem [Sem77] to automata over real vectors.

## References

- [AY07] Avigad, J., Yin, Y.: Quantifier elimination for the reals with a predicate for the powers of two. *Theoretical Computer Science* 370, 48–59 (2007)
- [BBR97] Boigelot, B., Bronne, L., Rassart, S.: An improved reachability analysis method for strongly linear hybrid systems. In: Grumberg, O. (ed.) *CAV 1997*. LNCS, vol. 1254, pp. 167–177. Springer, Heidelberg (1997)
- [Büc62] Büchi, J.R.: On a decision method in restricted second order arithmetic. In: *Proc. International Congress on Logic, Methodology and Philosophy of Science*, Stanford, pp. 1–12. Stanford University Press, Stanford (1962)
- [BHMV94] Bruyère, V., Hansel, G., Michaux, C., Villemaire, R.: Logic and  $p$ -recognizable sets of integers. *Bulletin of the Belgian Mathematical Society* 1(2), 191–238 (1994)
- [BJW05] Boigelot, B., Jodogne, S., Wolper, P.: An effective decision procedure for linear arithmetic over the integers and reals. *ACM Transactions on Computational Logic* 6(3), 614–633 (2005)
- [Boi98] Boigelot, B.: Symbolic methods for exploring infinite state spaces. PhD thesis, Université de Liège (1998)
- [Bru06] Brusten, J.: Etude des propriétés des RVA. Graduate thesis, Université de Liège (May 2006)
- [BRW98] Boigelot, B., Rassart, S., Wolper, P.: On the expressiveness of real and integer arithmetic automata. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) *ICALP 1998*. LNCS, vol. 1443, pp. 152–163. Springer, Heidelberg (1998)
- [Cob69] Cobham, A.: On the base-dependence of sets of numbers recognizable by finite automata. *Mathematical Systems Theory* 3, 186–192 (1969)
- [KV05] Kupferman, O., Vardi, M.Y.: Complementation constructions for nondeterministic automata on infinite words. In: Halbwachs, N., Zuck, L.D. (eds.) *TACAS 2005*. LNCS, vol. 3440, pp. 206–221. Springer, Heidelberg (2005)
- [Lat05] Latour, L.: Presburger arithmetic: from automata to formulas. PhD thesis, Université de Liège (2005)
- [Ler05] Leroux, J.: A polynomial time Presburger criterion and synthesis for number decision diagrams. In: *Proc. 20th LICS*, Chicago, June 2005, pp. 147–156. IEEE Computer Society Press, Los Alamitos (2005)
- [Sem77] Semenov, A.L.: Presburger-ness of predicates regular in two number systems. *Siberian Mathematical Journal* 18, 289–299 (1977)
- [vdD85] van den Dries, L.: The field of reals with a predicate for the powers of two. *Manuscripta Mathematica* 54, 187–195 (1985)
- [Vil92] Villemaire, R.: The theory of  $\langle \mathbb{N}, +, V_k, V_l \rangle$  is undecidable. *Theoretical Computer Science* 106(2), 337–349 (1992)
- [WB95] Wolper, P., Boigelot, B.: An automata-theoretic approach to Presburger arithmetic constraints. In: Mycroft, A. (ed.) *SAS 1995*. LNCS, vol. 983, Springer, Heidelberg (1995)
- [WB98] Wolper, P., Boigelot, B.: Verifying systems with infinite but regular state spaces. In: Vardi, M.Y. (ed.) *CAV 1998*. LNCS, vol. 1427, pp. 88–97. Springer, Heidelberg (1998)
- [Wil93] Wilke, T.: Locally threshold testable languages of infinite words. In: Enjalbert, P., Wagner, K.W., Finkel, A. (eds.) *STACS 93*. LNCS, vol. 665, pp. 607–616. Springer, Heidelberg (1993)



### A Proof of Lemma 1

For a base  $r > 1$  and a word  $w \in A_r \cdot \Sigma_r^\omega$ , let  $[w]_r$  denote the real number encoded by  $w$  in that base. Similarly, for  $w \in \{0, r - 1\} \cdot \Sigma_r^*$ , let  $[w]_r$  denote the integer number encoded by  $w$ , i.e.,  $[w]_r = [w \cdot \star \cdot 0^\omega]_r$ . For every  $k \geq 0$ , we define  $x_k = [w_1 \cdot (w_2)^k \cdot w_3 \cdot (w_4)^\omega]_{r_1}$ .

The prefix  $w_1$  can be decomposed into  $w_1 = w'_1 \cdot \star \cdot w''_1$ , with  $w'_1 \in \{0, r_1 - 1\} \cdot \Sigma_{r_1}^*$  and  $w''_1 \in \Sigma_{r_1}^*$ . We then have for every  $k > 0$ ,

$$x_k = \frac{y_k}{r_1^{|w''_1|+k|w_2|+|w_3|} (r_1^{|w_4|} - 1)}, \tag{1}$$

with  $y_k = (r_1^{|w_4|} - 1)[w'_1 \cdot w''_1 \cdot w_2^k \cdot w_3]_{r_1} + [0 \cdot w_4]_{r_1}$ . Remark that  $y_k$  is an integer, but cannot be a multiple of  $r_1$ . Indeed, we have  $y_k \bmod r_1 = ([0 \cdot w_4]_{r_1} - [w'_1 \cdot w''_1 \cdot w_2^k \cdot w_3]_{r_1}) \bmod r_1$ , which is non-zero thanks to the hypothesis on the last digits of  $w_2 \cdot w_3$  and  $w_4$ . For every  $k > 0$ , we have

$$y_k = \frac{z_k}{r_1^{|w_2|} - 1},$$

with  $z_k = ar_1^{k|w_2|} + b$ ,  $a = r_1^{|w_3|} (r_1^{|w_4|} - 1) ((r_1^{|w_2|} - 1)[w'_1 \cdot w''_1]_{r_1} + [0 \cdot w_2]_{r_1})$ , and  $b = -r_1^{|w_3|} (r_1^{|w_4|} - 1)[0 \cdot w_2]_{r_1} + (r_1^{|w_2|} - 1)(r_1^{|w_4|} - 1)[0 \cdot w_3]_{r_1} + (r_1^{|w_2|} - 1)[0 \cdot w_4]_{r_1}$ .

Substituting in (1), we get

$$x_k = \frac{z_k}{r_1^{|w''_1|+k|w_2|+|w_3|} (r_1^{|w_2|} - 1)(r_1^{|w_4|} - 1)}. \tag{2}$$

Since  $z_k = (r_1^{|w_2|} - 1)y_k$  and  $y_k \bmod r_1 \neq 0$ , we have  $z_k \bmod r_1 \neq 0$ , hence  $b \neq 0$ . Consider a prime factor  $f$  of  $r_1$ , and define  $l$  as the greatest integer such that  $f^l$  divides  $b$ . For every  $k > l$ , we have  $z_k \bmod f^l = 0$  and  $z_k \bmod f^{l+1} = b \bmod f^{l+1} \neq 0$ . It follows that the reduced rational expression of  $x_k$ , i.e.,  $x_k = n_k/d_k$  with  $n_k, d_k \in \mathbb{Z}$ ,  $d_k > 0$  and  $\gcd(n_k, d_k) = 1$ , is such that  $f^{k-l}$  divides  $d_k$  for every  $k > l$ . Indeed, the numerator of (2) is not divisible by  $f^{l+1}$  whereas its denominator is divisible by  $f^{k+1}$ .

Assume now, by contradiction, that the set  $\{x_k \mid k \geq 0\}$  can be represented in base  $r_2$  using only a finite number of ultimate periods. Then, there exists an ultimate period  $v \in \Sigma_{r_2}^+$  such that for infinitely many values of  $k$ , we have  $x_k = [u'_k \cdot \star \cdot u''_k \cdot v^\omega]_{r_2}$ , with  $u'_k \in \{0, r_2 - 1\} \cdot \Sigma_{r_2}^*$  and  $u''_k \in \Sigma_{r_2}^*$ . We then have, for these values of  $k$ ,

$$x_k = \frac{[u'_k \cdot u''_k \cdot v]_{r_2} - [u'_k \cdot u''_k]_{r_2}}{r_2^{|u''_k|} (r_2^{|v|} - 1)}.$$

Since  $(r_2^{|v|} - 1)$  is bounded, and  $r_2$  is relatively prime with  $r_1$  by hypothesis, the denominator of this expression can only be divisible by a bounded number of powers of  $f$ , which contradicts our previous result. □

# Minimum-Time Reachability in Timed Games<sup>\*</sup>

Thomas Brihaye<sup>1</sup>, Thomas A. Henzinger<sup>2</sup>, Vinayak S. Prabhu<sup>3</sup>,  
and Jean-François Raskin<sup>4</sup>

<sup>1</sup> LSV-CNRS & ENS de Cachan

`thomas.brihaye@lsv.ens-cachan.fr`

<sup>2</sup>Department of Computer and Communication Sciences, EPFL

`tah@epfl.ch`

<sup>3</sup>Department of Electrical Engineering & Computer Sciences, UC Berkeley

`vinayak@eecs.berkeley.edu`

<sup>4</sup>Département d'Informatique, Université Libre de Bruxelles

`jraskin@ulb.ac.be`

**Abstract.** We consider the minimum-time reachability problem in concurrent two-player timed automaton game structures. We show how to compute the minimum time needed by a player to reach a target location against all possible choices of the opponent. We do not put any syntactic restriction on the game structure, nor do we require any player to guarantee time divergence. We only require players to use receptive strategies which do not block time. The minimal time is computed in part using a fixpoint expression, which we show can be evaluated on equivalence classes of a non-trivial extension of the clock-region equivalence relation for timed automata.

## 1 Introduction

*Timed automata* [3], finite-state machines enriched with clocks and clock constraints, are a well-established formalism for the modeling and analysis of timed systems. A large number of important and interesting theoretical results have been obtained on problems in the timed automaton framework. In parallel with these theoretical results, efficient verification tools have been implemented and successfully applied to industrially relevant case studies.

Timed automata are models for closed systems, where every transition is controlled. If we want to distinguish between actions of several agents (for instance, a *controller* and an *environment*), we have to consider games on timed automata, also known as *timed automaton games*. In the sequel, we will focus on two-player games. In this context, the *reachability problem* asks whether player-1 has a strategy to force the timed game to reach a target location, no matter how player-2 resolves her choices. These games were first introduced and studied in [22,19]. In this framework, it is also natural to consider the *minimum-time reachability problem*, which asks for the minimal time required by player-1 to reach a target

---

<sup>\*</sup> This research was supported in part by the NSF grant CCR-0225610 and by the Swiss National Science Foundation.

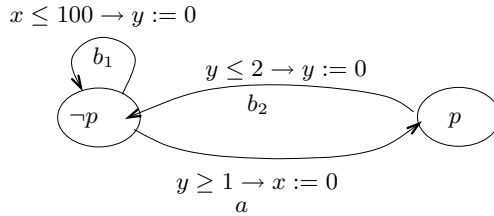


Fig. 1. A timed automaton game

location, no matter how player-2 resolves her choices. This problem was first posed in [5], where it was shown to be decidable for a restricted class of timed automaton games.

Any formalism involving timed systems has to face the problem of *zeno runs*, i.e. runs of the model where time converges. Zeno runs are not physically meaningful. The avoidance of such runs has often been achieved by putting syntactic constraints on the cycles of timed automaton games [20,5,15,6], or by semantic conditions that discretize time [16]. Other works on the existence of controllers [19,14,7,9] have required that time divergence be ensured by the controller —a one-sided, unfair view in settings where the player modeling the environment can also block time.

Recently, a more equitable treatment of zeno runs has been proposed in [12]. This setting formulates a symmetric set-up of the model, where both players are given equally powerful options for updating the state of the game, advancing time, or blocking time. Both players may block time, however, for a player to win for an objective, she must not be *responsible* for preventing time from diverging. It has been shown in [17] that this is equivalent to requiring that the players use only *receptive* strategies [4,21], which do not prevent time from diverging.

**Example.** Consider the game depicted in Figure 1. Let edge *a* be controlled by player-1; the others being controlled by player-2. Suppose we want to know what is the earliest time that player-1 can reach *p* starting from the state  $\langle -p, x = 0, y = 0 \rangle$  (i.e., the initial values of both clocks *x* and *y* are 0). Player-1 is not able to guarantee time divergence, as player-2 can keep on choosing the edge *b*<sub>1</sub>. On the other hand, we do not want to put any restriction of the number of times that player-2 chooses *b*<sub>1</sub>. Requiring that the players use only non-zeno strategies avoids such unnecessary restrictions, and gives the correct minimum time for player-1 to reach *p*, namely, 101 time units.

**Contribution.** We consider the *minimum-time reachability problem (for timed automaton games)*, in the framework of [12,17]. We present an EXPTIME algorithm to compute the minimum time needed by player-1 to force the game into a target location, with both players restricted to using only receptive strategies (note that reachability in timed automaton games is EXPTIME-complete [16]). The proof technique builds on techniques from [12,17]. We first show that the minimum time can be obtained by solving a certain  $\mu$ -calculus fixpoint equation. We then give a proof of termination for the fixpoint evaluation. This requires

an important new ingredient: an extension of the clock-region equivalence [3] for timed automata. We show our extended region equivalence classes to be stable with respect to the monotone functions used in the fixpoint equation. Using results from [17], we manage to restrict the fixpoint computation to finitely many regions and thus guarantee termination.

We note that standard clock regions do not suffice for the solution. The minimum-time reachability game has two components: a reachability part that can be handled by discrete arguments based on the clock-region graph; and a minimum-time part that requires minimization within clock regions (cf. [11]). Unfortunately, both arguments are intertwined and cannot be considered in isolation. Our extended regions decouple the two parts in the proofs. We also note that region sequences that correspond to time-minimal runs may in general be required to contain region cycles in which time does not progress by an integer amount; thus a reduction to a loop-free region game, as in [1], is not possible.

**Related work.** Only special cases of the minimum-time reachability problem have been solved before: [5] restricts attention to the case where every cycle of the timed automaton ensures syntactically that a positive amount of time passes (a.k.a. strong non-zenoness assumption); [2] considers timed automaton games that are restricted to a bounded number of moves; [17] presents an approximate computation of the minimum time (computation of the exact minimum time being left open). The general case for *weighted* timed automaton games (timed automaton games augmented with costs on discrete transitions and cost rates on locations) is undecidable [8]. The recent work of [18] presents a strategy improvement algorithm that computes the minimum time in all timed automaton games, but it does not require strategies to be receptive. Average-reward games in the framework of [12] are considered in [1], but with the durations of time moves restricted to either 0 or 1. The non-game version of the minimum-time reachability problem is solved in [11].

**Outline.** In Section 2, we recall the definitions of the timed games framework from [12]. The minimum-time reachability problem is defined in Section 3. Section 4 gives an algorithm that computes the minimum time in timed automaton games. The algorithm runs in time exponential in the number of clocks and the size of clock constraints.

## 2 Timed Games

### 2.1 Timed Game Structures

We use the formalism of [12]. A *timed game structure* is a tuple  $\mathcal{G} = \langle S, \Sigma, \sigma, A_1, A_2, \Gamma_1, \Gamma_2, \delta \rangle$  with the following components:

- $S$  is a set of states.
- $\Sigma$  is a finite set of propositions.
- $\sigma : S \mapsto 2^\Sigma$  is the observation map, which assigns to every state the set of propositions that are true in that state.

- $A_1$  and  $A_2$  are two disjoint sets of actions for players 1 and 2, respectively. We assume that  $\perp_i \notin A_i$ , and write  $A_i^\perp$  for  $A_i \cup \{\perp_i\}$ . We also assume  $A_1^\perp$  and  $A_2^\perp$  to be disjoint. The set of *moves* for player  $i$  is  $M_i = \mathbb{R}_{\geq 0} \times A_i^\perp$ . Intuitively, a move  $\langle \Delta, a_i \rangle$  by player  $i$  indicates a waiting period of  $\Delta$  time units followed by a discrete transition labeled with action  $a_i$ .
- $\Gamma_i : S \mapsto 2^{M_i} \setminus \emptyset$  are two move assignments. At every state  $s$ , the set  $\Gamma_i(s)$  contains the moves that are available to player  $i$ . We require that  $\langle 0, \perp_i \rangle \in \Gamma_i(s)$  for all states  $s \in S$  and  $i \in \{1, 2\}$ . Intuitively,  $\langle 0, \perp_i \rangle$  is a time-blocking stutter move.
- $\delta : S \times (M_1 \cup M_2) \mapsto S$  is the transition function. We require that for all time delays  $\Delta, \Delta' \in \mathbb{R}_{\geq 0}$  with  $\Delta' \leq \Delta$ , and all actions  $a_i \in A_i^\perp$ , we have (1)  $\langle \Delta, a_i \rangle \in \Gamma_i(s)$  iff both  $\langle \Delta', \perp_i \rangle \in \Gamma_i(s)$  and  $\langle \Delta - \Delta', a_i \rangle \in \Gamma_i(\delta(s, \langle \Delta', \perp_i \rangle))$ ; and (2) if  $\delta(s, \langle \Delta', \perp_i \rangle) = s'$  and  $\delta(s', \langle \Delta - \Delta', a_i \rangle) = s''$ , then  $\delta(s, \langle \Delta, a_i \rangle) = s''$ .

The game proceeds as follows. If the current state of the game is  $s$ , then both players simultaneously propose moves  $\langle \Delta_1, a_1 \rangle \in \Gamma_1(s)$  and  $\langle \Delta_2, a_2 \rangle \in \Gamma_2(s)$ . The move with the shorter duration “wins” in determining the next state of the game. If both moves have the same duration, then one of the two moves is chosen non-deterministically. Formally, we define the *joint destination function*  $\delta_{\text{jd}} : S \times M_1 \times M_2 \mapsto 2^S$  by

$$\delta_{\text{jd}}(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle) = \begin{cases} \{\delta(s, \langle \Delta_1, a_1 \rangle)\} & \text{if } \Delta_1 < \Delta_2; \\ \{\delta(s, \langle \Delta_2, a_2 \rangle)\} & \text{if } \Delta_2 < \Delta_1; \\ \{\delta(s, \langle \Delta_1, a_1 \rangle), \delta(s, \langle \Delta_2, a_2 \rangle)\} & \text{if } \Delta_1 = \Delta_2. \end{cases}$$

The time elapsed when the moves  $m_1 = \langle \Delta_1, a_1 \rangle$  and  $m_2 = \langle \Delta_2, a_2 \rangle$  are proposed is given by  $\text{delay}(m_1, m_2) = \min(\Delta_1, \Delta_2)$ . The boolean predicate  $\text{blame}_i(s, m_1, m_2, s')$  indicates whether player  $i$  is “responsible” for the state change from  $s$  to  $s'$  when the moves  $m_1$  and  $m_2$  are proposed. Denoting the opponent of player  $i \in \{1, 2\}$  by  $\sim i = 3 - i$ , we define

$$\text{blame}_i(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle, s') = (\Delta_i \leq \Delta_{\sim i} \wedge \delta(s, \langle \Delta_i, a_i \rangle) = s').$$

A *run* of the timed game structure  $\mathcal{G}$  is an infinite sequence  $r = s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \dots$  such that  $s_k \in S$  and  $m_i^k \in \Gamma_i(s_k)$  and  $s_{k+1} \in \delta_{\text{jd}}(s_k, m_1^k, m_2^k)$  for all  $k \geq 0$  and  $i \in \{1, 2\}$ . For  $k \geq 0$ , let  $\text{time}(r, k)$  denote the “time” at position  $k$  of the run, namely,  $\text{time}(r, k) = \sum_{j=0}^{k-1} \text{delay}(m_1^j, m_2^j)$  (we let  $\text{time}(r, 0) = 0$ ). By  $r[k]$  we denote the  $(k + 1)$ -th state  $s_k$  of  $r$ . The run prefix  $r[0..k]$  is the finite prefix of the run  $r$  that ends in the state  $s_k$ ; we write  $\text{last}(r[0..k])$  for the ending state  $s_k$  of the run prefix. Let  $\text{Runs}$  be the set of all runs of  $\mathcal{G}$ , and let  $\text{FinRuns}$  be the set of run prefixes.

A *strategy*  $\pi_i$  for player  $i \in \{1, 2\}$  is a function  $\pi_i : \text{FinRuns} \mapsto M_i$  that assigns to every run prefix  $r[0..k]$  a move to be proposed by player  $i$  at the state  $\text{last}(r[0..k])$  if the history of the game is  $r[0..k]$ . We require that  $\pi_i(r[0..k]) \in \Gamma_i(\text{last}(r[0..k]))$  for every run prefix  $r[0..k]$ , so that strategies propose only available moves. The results of this paper are equally valid if strategies

do not depend on past moves chosen by the players, but only on the past sequence of states and time delays [12]. For  $i \in \{1, 2\}$ , let  $\Pi_i$  be the set of player- $i$  strategies. Given two strategies  $\pi_1 \in \Pi_1$  and  $\pi_2 \in \Pi_2$ , the set of possible *outcomes* of the game starting from a state  $s \in S$  is denoted  $\text{Outcomes}(s, \pi_1, \pi_2)$ : it contains all runs  $r = s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \dots$  such that  $s_0 = s$  and for all  $k \geq 0$  and  $i \in \{1, 2\}$ , we have  $\pi_i(r[0..k]) = m_i^k$ .

We distinguish between *physical time* and *game time*. We allow moves with zero time delay, thus a physical time  $t \in \mathbb{R}_{\geq 0}$  may correspond to several linearly ordered states, to which we assign the game times  $\langle t, 0 \rangle, \langle t, 1 \rangle, \langle t, 2 \rangle, \dots$ . For a run  $r \in \text{Runs}$ , we define the set of game times as

$$\text{GameTimes}(r) = \{ \langle t, k \rangle \in \mathbb{R}_{\geq 0} \times \mathbb{N} \mid 0 \leq k < |\{j \geq 0 \mid \text{time}(r, j) = t\}| \} \cup \{ \langle t, 0 \rangle \mid \text{time}(r, j) \geq t \text{ for some } j \geq 0 \}.$$

The state of the run  $r$  at a game time  $\langle t, k \rangle \in \text{GameTimes}(r)$  is defined as

$$\text{state}(r, \langle t, k \rangle) = \begin{cases} r[j+k] & \text{if } \text{time}(r, j) = t \text{ and for all } j' < j, \text{time}(r, j') < t; \\ \delta(r[j], \langle t - \text{time}(r, j), \perp_i \rangle) & \text{if } \text{time}(r, j) < t < \text{time}(r, j+1) \text{ and} \\ & r[0..j+1] = r[0..j], \langle m_1^j, m_2^j \rangle, r[j+1] \text{ and} \\ & \text{blame}_i(r[j], m_1^j, m_2^j, r[j+1]) \end{cases}$$

Note that if  $r$  is a run of the timed game structure  $\mathcal{G}$ , and  $\text{time}(r, j) < t < \text{time}(r, j+1)$ , then  $\delta(r[j], \langle t - \text{time}(r, j), \perp_i \rangle)$  is a state in  $S$ , namely, the state that results from  $r[j]$  by letting time  $t - \text{time}(r, j)$  pass. We say that the run  $r$  *visits* a set  $X \subseteq S$  at time  $t$  if there is a  $\tau = \langle t, k \rangle \in \text{GameTimes}(r)$  such that  $\text{state}(r, \tau) \in X$ . A run  $r$  visits a proposition  $p \in \Sigma$  if it visits the set  $S_p$  defined as  $\{s \mid p \in \sigma(s)\}$ .

### 2.2 Timed Automaton Games

Timed automata [3] suggest a finite syntax for specifying infinite-state timed game structures. A *timed automaton game* is a tuple  $\mathcal{T} = \langle L, \Sigma, \sigma, C, A_1, A_2, E, \gamma \rangle$  with the following components:

- $L$  is a finite set of locations.
- $\Sigma$  is a finite set of propositions.
- $\sigma : L \mapsto 2^\Sigma$  assigns to every location a set of propositions.
- $C$  is a finite set of clocks. We assume that  $z \in C$  for the unresettable clock  $z$ , which is used to measure the time elapsed since the start of the game.
- $A_1$  and  $A_2$  are two disjoint sets of actions for players 1 and 2, respectively.
- $E \subseteq L \times (A_1 \cup A_2) \times \text{Constr}(C) \times L \times 2^{C \setminus \{z\}}$  is the edge relation, where the set  $\text{Constr}(C)$  of *clock constraints* is generated by the grammar

$$\theta ::= x \leq d \mid d \leq x \mid \neg\theta \mid \theta_1 \wedge \theta_2$$

for clock variables  $x \in C$  and nonnegative integer constants  $d$ . For an edge  $e = \langle l, a_i, \theta, l', \lambda \rangle$ , the clock constraint  $\theta$  acts as a guard on the clock values which specifies when the edge  $e$  can be taken, and by taking the edge  $e$ , the

clocks in the set  $\lambda \subseteq C \setminus \{z\}$  are reset to 0. We require that for all edges  $\langle l, a_i, \theta', l', \lambda' \rangle, \langle l, a_i, \theta'', l'', \lambda'' \rangle \in E$  with  $l' \neq l''$ , the conjunction  $\theta' \wedge \theta''$  is unsatisfiable. This requirement ensures that a state and a move together uniquely determine a successor state.

- $\gamma : L \mapsto \text{Constr}(C)$  is a function that assigns to every location an invariant for both players. All clocks increase uniformly at the same rate. When at location  $l$ , each player  $i$  must propose a move out of  $l$  before the invariant  $\gamma(l)$  expires. Thus, the game can stay at a location only as long as the invariant is satisfied by the clock values.

A *clock valuation* is a function  $\kappa : C \mapsto \mathbb{R}_{\geq 0}$  that maps every clock to a non-negative real. The set of all clock valuations for  $C$  is denoted by  $K(C)$ . Given a clock valuation  $\kappa \in K(C)$  and a time delay  $\Delta \in \mathbb{R}_{\geq 0}$ , we write  $\kappa + \Delta$  for the clock valuation in  $K(C)$  defined by  $(\kappa + \Delta)(x) = \kappa(x) + \Delta$  for all clocks  $x \in C$ . For a subset  $\lambda \subseteq C$  of the clocks, we write  $\kappa[\lambda := 0]$  for the clock valuation in  $K(C)$  defined by  $(\kappa[\lambda := 0])(x) = 0$  if  $x \in \lambda$ , and  $(\kappa[\lambda := 0])(x) = \kappa(x)$  if  $x \notin \lambda$ . A clock valuation  $\kappa \in K(C)$  *satisfies* the clock constraint  $\theta \in \text{Constr}(C)$ , written  $\kappa \models \theta$ , if the condition  $\theta$  holds when all clocks in  $C$  take on the values specified by  $\kappa$ .

A *state*  $s = \langle l, \kappa \rangle$  of the timed automaton game  $\mathcal{T}$  is a location  $l \in L$  together with a clock valuation  $\kappa \in K(C)$  such that the invariant at the location is satisfied, that is,  $\kappa \models \gamma(l)$ . Let  $S$  be the set of all states of  $\mathcal{T}$ . In a state, each player  $i$  proposes a time delay allowed by the invariant map  $\gamma$ , together either with the action  $\perp$ , or with an action  $a_i \in A_i$  such that an edge labeled  $a_i$  is enabled after the proposed time delay. We require that for  $i \in \{1, 2\}$  and for all states  $s = \langle l, \kappa \rangle$ , if  $\kappa \models \gamma(l)$ , either  $\kappa + \Delta \models \gamma(l)$  for all  $\Delta \in \mathbb{R}_{\geq 0}$ , or there exist a time delay  $\Delta \in \mathbb{R}_{\geq 0}$  and an edge  $\langle l, a_i, \theta, l', \lambda \rangle \in E$  such that (1)  $a_i \in A_i$  and (2)  $\kappa + \Delta \models \theta$  and for all  $0 \leq \Delta' \leq \Delta$ , we have  $\kappa + \Delta' \models \gamma(l)$ , and (3)  $(\kappa + \Delta)[\lambda := 0] \models \gamma(l')$ .

The timed automaton game  $\mathcal{T}$  defines the following timed game structure  $\llbracket \mathcal{T} \rrbracket = \langle S, \Sigma, \sigma^*, A_1, A_2, \Gamma_1, \Gamma_2, \delta \rangle$ :

- $S$  is defined above.
- $\sigma^*(\langle l, \kappa \rangle) = \sigma(l)$ .
- For  $i \in \{1, 2\}$ , the set  $\Gamma_i(\langle l, \kappa \rangle)$  contains the following elements:
  1.  $\langle \Delta, \perp_i \rangle$  if for all  $0 \leq \Delta' \leq \Delta$ , we have  $\kappa + \Delta' \models \gamma(l)$ .
  2.  $\langle \Delta, a_i \rangle$  if for all  $0 \leq \Delta' \leq \Delta$ , we have  $\kappa + \Delta' \models \gamma(l)$ , and  $a_i \in A_i$ , and there exists an edge  $\langle l, a_i, \theta, l', \lambda \rangle \in E$  such that  $\kappa + \Delta \models \theta$ .
- $\delta(\langle l, \kappa \rangle, \langle \Delta, \perp_i \rangle) = \langle l, \kappa + \Delta \rangle$ , and  $\delta(\langle l, \kappa \rangle, \langle \Delta, a_i \rangle) = \langle l', (\kappa + \Delta)[\lambda := 0] \rangle$  for the unique edge  $\langle l, a_i, \theta, l', \lambda \rangle \in E$  with  $\kappa + \Delta \models \theta$ .

### 2.3 Clock Regions

Timed automaton games can be solved using a region construction from the theory of timed automata [B]. For a real  $t \geq 0$ , let  $\text{frac}(t) = t - \lfloor t \rfloor$  denote the fractional part of  $t$ . Given a timed automaton game  $\mathcal{T}$ , for each clock  $x \in C$ ,

let  $c_x$  denote the largest integer constant that appears in any clock constraint involving  $x$  in  $\mathcal{T}$ . Two clock valuations  $\kappa_1, \kappa_2 \in K(C)$  are *clock-region equivalent*, denoted  $\kappa_1 \cong \kappa_2$ , if the following three conditions hold:

1. For all  $x \in C$ , either  $\lfloor \kappa_1(x) \rfloor = \lfloor \kappa_2(x) \rfloor$ , or both  $\lfloor \kappa_1(x) \rfloor > c_x$ ,  $\lfloor \kappa_2(x) \rfloor > c_x$ .
2. For all  $x, y \in C$  with  $\kappa_1(x) \leq c_x$  and  $\kappa_1(y) \leq c_y$ , we have  $\text{frac}(\kappa_1(x)) \leq \text{frac}(\kappa_1(y))$  iff  $\text{frac}(\kappa_2(x)) \leq \text{frac}(\kappa_2(y))$ .
3. For all  $x \in C$  with  $\kappa_1(x) \leq c_x$ , we have  $\text{frac}(\kappa_1(x)) = 0$  iff  $\text{frac}(\kappa_2(x)) = 0$ .

Two states  $\langle l_1, \kappa_1 \rangle, \langle l_2, \kappa_2 \rangle \in S$  are *clock-region equivalent*, denoted  $\langle l_1, \kappa_1 \rangle \cong \langle l_2, \kappa_2 \rangle$ , iff  $l_1 = l_2$  and  $\kappa_1 \cong \kappa_2$ . It is not difficult to see that  $\cong$  is an equivalence relation on  $S$ . A *clock region* is an equivalence class with respect to  $\cong$ . There are finitely many clock regions; more precisely, the number of clock regions is bounded by  $|L| \cdot \prod_{x \in C} (c_x + 1) \cdot |C|! \cdot 2^{|C|}$ . For a state  $s \in S$ , we write  $[s] \subseteq S$  for the clock region containing  $s$ . These clock regions induce a time-abstract bisimulation.

### 3 The Minimum-Time Reachability Problem

Given a state  $s$  and a target proposition  $p \in \Sigma$  in a timed game structure  $\mathcal{G}$ , the *reachability* problem is to determine whether starting from  $s$ , player-1 has a strategy for visiting the proposition  $p$ . We must make sure that player-2 does not prevent player-1 from reaching a target state by blocking time. We also require player-1 to not block time as it can lead to physically unmeaningful plays. These requirements can be achieved by requiring strategies to be *receptive* [21,4]. Formally, we first define the following two sets of runs:

- $\text{Timediv} \subseteq \text{Runs}$  is the set of all time-divergent runs. A run  $r$  is *time-divergent* if  $\lim_{k \rightarrow \infty} \text{time}(r, k) = \infty$ .
- $\text{Blameless}_i \subseteq \text{Runs}$  is the set of runs in which player  $i$  is responsible only for finitely many transitions. A run  $s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \dots$  belongs to the set  $\text{Blameless}_i$ , for  $i = \{1, 2\}$ , if there exists a  $k \geq 0$  such that for all  $j \geq k$ , we have  $\neg \text{blame}_i(s_j, m_1^j, m_2^j, s_{j+1})$ .

A strategy  $\pi_i$  for player  $i \in \{1, 2\}$  is *receptive* if for all opposing strategies  $\pi_{\sim i}$ , and all states  $s \in S$ ,  $\text{Outcomes}(s, \pi_1, \pi_2) \subseteq \text{Timediv} \cup \text{Blameless}_i$ . Thus, no what matter what the opponent does, a receptive player- $i$  strategy should not be responsible for blocking time. Strategies that are not receptive are not physically meaningful (note that receptiveness is not sufficient for a strategy to be physically meaningful, see [10]). For  $i \in \{1, 2\}$ , let  $\Pi_i^R$  be the set of player- $i$  receptive strategies. A timed game structure is *well-formed* if both players have receptive strategies. We restrict our attention to well-formed timed game structures. Well-formedness of timed automaton games can be checked for (see [17]).

We say player-1 *wins* for the reachability objective  $p$  at state  $s$ , denoted  $s \in \langle\langle 1 \rangle\rangle \diamond p$ , if he has a receptive strategy  $\pi_1$  such that for all player-2 receptive strategies  $\pi_2$ , we have that all runs  $r \in \text{Outcomes}(s, \pi_1, \pi_2)$  visit  $p$ .



Equivalently [17], we can define player-1 to be winning for the reachability objective  $p$  at state  $s$  if he has a strategy  $\pi_1$  such that for all player-2 strategies  $\pi_2$ , for all runs  $r \in \text{Outcomes}(s, \pi_1, \pi_2)$ :

- if  $r \in \text{Timediv}$ , then  $r$  visits the proposition  $p$ ;
- if  $r \notin \text{Timediv}$ , then  $r \in \text{Blameless}_1$ .

The *minimum-time reachability problem* is to determine the minimal time in which a player can force the game into a set of target states, using only receptive strategies. Formally, given a timed game structure  $\mathcal{G}$ , a target proposition  $p \in \Sigma$ , and a run  $r$  of  $\mathcal{G}$ , let

$$T_{\text{visit}}(\mathcal{G}, r, p) = \begin{cases} \infty & \text{if } r \text{ does not visit } p; \\ \inf \{t \in \mathbb{R}_{\geq 0} \mid p \in \sigma(\text{state}(r, \langle t, k \rangle)) \text{ for some } k\} & \text{otherwise.} \end{cases}$$

The *minimal time* for player-1 to force the game from a start state  $s \in S$  to a visit to  $p$  is then

$$T_{\min}(\mathcal{G}, s, p) = \inf_{\pi_1 \in \Pi_1^R} \sup_{\pi_2 \in \Pi_2^R} \sup_{r \in \text{Outcomes}(s, \pi_1, \pi_2)} T_{\text{visit}}(\mathcal{G}, r, p)$$

We omit  $\mathcal{G}$  when clear from the context.

## 4 Solving for Minimum-Time Reachability

We restrict our attention to well-formed timed automaton games. The definition of  $T_{\min}$  quantifies strategies over the set of receptive strategies. Our algorithm will instead work over the set of *all* strategies. Theorem 1 presents this reduction. We will then present a game structure for the timed automaton game  $\mathcal{T}$  in which  $\text{Timediv}$  and  $\text{Blameless}_1$  can be represented using Büchi and co-Büchi constraints. This builds on the framework of [12] in which a run satisfies the reachability objective  $p$  for player-1 iff it belongs in  $(\text{Timediv} \cap \text{Reach}(p)) \cup (\neg \text{Timediv} \cap \text{Blameless}_1)$ , where  $\text{Reach}(p)$  denotes the set of runs which visit  $p$ . In addition, our game structure will also have a backwards running clock, which will be used in the computation of the minimum time, using a  $\mu$ -calculus algorithm on *extended regions*.

### 4.1 Allowing Players to Use All Strategies

To allow quantification over all strategies, we first modify the payoff function  $T_{\text{visit}}$ , so that players are maximally penalised on zeno runs:

$$T_{\text{visit}}^{\text{UR}}(r, p) = \begin{cases} \infty & \text{if } r \notin \text{Timediv} \text{ and } r \notin \text{Blameless}_i; \\ \infty & \text{if } r \in \text{Timediv} \text{ and } r \text{ does not visit } p; \\ 0 & \text{if } r \notin \text{Timediv} \text{ and } r \in \text{Blameless}_i; \\ \inf \{t \in \mathbb{R}_{\geq 0} \mid p \in \sigma(\text{state}(r, \langle t, k \rangle)) \text{ for some } k\} & \text{otherwise.} \end{cases}$$

It turns out that penalizing on zeno-runs is equivalent to penalising on non-receptive strategies:

**Theorem 1.** *Let  $s$  be a state and  $p$  a proposition in a well-formed timed game structure  $\mathcal{G}$ . Then:*

$$T_{\min}(s, p) = \inf_{\pi_1 \in \Pi_1} \sup_{\pi_2 \in \Pi_2} \sup_{r \in \text{Outcomes}(s, \pi_1, \pi_2)} T_{\text{visit}}^{\text{UR}}(r, p)$$

**4.2 Reduction to Reachability with Büchi and co-Büchi Constraints**

We now decouple reachability from optimizing for minimal time, and show how reachability with time divergence can be solved for, using an appropriately chosen  $\mu$ -calculus fixpoint.

**Lemma 1** ([17]). *Given a state  $s$ , and a proposition  $p$  of a well-formed timed automaton game  $\mathcal{T}$ , 1) we can determine if  $T_{\min}(s, p) < \infty$ , and 2) If  $T_{\min}(s, p) < \infty$ , then  $T_{\min}(s, p) < M = 8|L| \cdot \prod_{x \in C} (c_x + 1) \cdot |C + 1|! \cdot 2^{|C|}$ . This upper bound is the same for all  $s' \cong s$ .*

Let  $M$  be the upper bound on  $T_{\min}(s, p)$  as in Lemma 1 if  $T_{\min}(s, p) < \infty$ , and  $M = 1$  otherwise. For a number  $N$ , let  $\mathbb{R}_{[0, N]}$  and  $\mathbb{R}_{(0, N)}$  denote  $\mathbb{R} \cap [0, N]$  and  $\mathbb{R} \cap (0, N)$  respectively. We first look at the enlarged game structure  $\widehat{\mathbb{J}}$  with the state space  $\widehat{S} = S \times \mathbb{R}_{(0, 1)} \times (\mathbb{R}_{[0, M]} \cup \{\perp\}) \times \{\text{TRUE}, \text{FALSE}\}^2$ , and an augmented transition relation  $\widehat{\delta} : \widehat{S} \times (M_1 \cup M_2) \mapsto \widehat{S}$ . In an augmented state  $\langle s, \mathfrak{z}, \beta, \text{tick}, \text{bl}_1 \rangle \in \widehat{S}$ , the component  $s \in S$  is a state of the original game structure  $\mathbb{J}$ ,  $\mathfrak{z}$  is value of a fictitious clock  $z$  which gets reset every time it hits 1,  $\beta$  is the value of a fictitious clock which is running *backwards*, *tick* is true iff the last transition resulted in the clock  $z$  hitting 1 (so *tick* is true iff the last transition resulted in  $\mathfrak{z} = 0$ ), and  $\text{bl}_1$  is true if player-1 is to blame for the last transition.

Formally,  $\langle s', \mathfrak{z}', \beta', \text{tick}', \text{bl}'_1 \rangle = \widehat{\delta}(\langle s, \mathfrak{z}, \beta, \text{tick}, \text{bl}_1 \rangle, \langle \Delta, a_i \rangle)$  iff

1.  $s' = \delta(s, \langle \Delta, a_i \rangle)$
2.  $\mathfrak{z}' = (\mathfrak{z} + \Delta) \bmod 1$ ;
3.  $\beta' = \beta \ominus \Delta$ , where we define  $\beta \ominus \Delta$  as  $\beta - \Delta$  if  $\beta \neq \perp$  and  $\beta - \Delta \geq 0$ , and  $\perp$  otherwise ( $\perp$  is an absorbing value for  $\beta$ ).
4.  $\text{tick}' = \text{TRUE}$  if  $\mathfrak{z} + \Delta \geq 1$ , and  $\text{FALSE}$  otherwise
5.  $\text{bl}'_1 = \text{TRUE}$  if  $a_i \in A_1^\perp$  and  $\text{FALSE}$  otherwise.

Each run  $r$  of  $\mathbb{J}$ , and values  $\mathfrak{z} \in \mathbb{R}_{\geq 0}, \beta \leq M$  can be mapped to a corresponding unique run  $\widehat{r}_{\mathfrak{z}, \beta}$  in  $\widehat{\mathbb{J}}$ , with  $\widehat{r}_{\mathfrak{z}, \beta}[0] = \langle r[0], \mathfrak{z}, \beta, \text{FALSE}, \text{FALSE} \rangle$ . Similarly, each run  $\widehat{r}$  of  $\widehat{\mathbb{J}}$  can be projected to a unique run  $\widehat{r} \downarrow \mathcal{T}$  of  $\mathbb{J}$ . It can be seen that the run  $r$  is in  $\text{Timediv}$  iff *tick* is true infinitely often in  $\widehat{r}_{\mathfrak{z}, \beta}$ , and that the set  $\text{Blameless}_1$  corresponds to runs along which  $\text{bl}_1$  is true only finitely often.

**Lemma 2.** *Given a timed game structure  $\mathbb{J}$ , let  $\widehat{X}_p = S_p \times \mathbb{R}_{(0, 1)} \times \{0\} \times \{\text{TRUE}, \text{FALSE}\}^2$ .*

1. *For a run  $r$  of the timed game structure  $\mathbb{J}$ , let  $T_{\text{visit}}(r, p) < \infty$ . Then,  $T_{\text{visit}}(r, p) = \inf\{\beta \mid \beta \in \mathbb{R}_{[0, M]} \text{ and } \widehat{r}_{0, \beta} \text{ visits the set } \widehat{X}_p\}$ .*

2. Let  $T_{\min}(s, p) < \infty$ . Then,

$$T_{\min}(s, p) = \inf \left\{ \beta \mid \beta \in \mathbb{R}_{[0, M]} \text{ and } \langle s, 0, \beta, \text{FALSE}, \text{FALSE} \rangle \in \langle\langle 1 \rangle\rangle \diamond \widehat{X}_p \right\}$$

3. If  $T_{\min}(s, p) = \infty$ , then for all  $\beta$ , we have  $\langle s, 0, \beta, \text{FALSE}, \text{FALSE} \rangle \notin \langle\langle 1 \rangle\rangle \diamond \widehat{X}_p$ .

The reachability objective can be reduced to a parity game: each state in  $\widehat{S}$  is assigned an index  $\Omega : \widehat{S} \mapsto \{0, 1\}$ , with  $\Omega(\widehat{s}) = 1$  iff  $\widehat{s} \notin \widehat{X}_p$ ; and  $\text{tick} \vee \text{bl}_1 = \text{TRUE}$ . We also modify the game structure so that the states in  $\widehat{X}_p$  are absorbing.

**Lemma 3.** *For the timed game  $\llbracket \mathcal{J} \rrbracket$  with the reachability objective  $\widehat{X}_p$ , the state  $\widehat{s} = \langle s, 0, \beta, \text{FALSE}, \text{FALSE} \rangle \in \langle\langle 1 \rangle\rangle \diamond \widehat{X}_p$  iff player-1 has a strategy  $\pi_1$  such that for all strategies  $\pi_2$  of player-2, and all runs  $\widehat{r}_{0, \beta} \in \text{Outcomes}(\widehat{s}, \pi_1, \pi_2)$ , the index 1 does not occur infinitely often in  $\widehat{r}_{0, \beta}$ .*

The fixpoint formula for solving the parity game in Lemma 3 is given by (as in [13]),

$$Y = \mu Y \nu Z \left[ (\Omega^{-1}(1) \cap \text{CPre}_1(Y)) \cup (\Omega^{-1}(0) \cap \text{CPre}_1(Z)) \right]$$

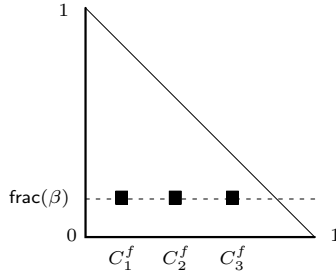
The fixpoint expression uses the variables  $Y, Z \subseteq \widehat{S}$  and the controllable predecessor operator,  $\text{CPre}_1 : 2^{\widehat{S}} \mapsto 2^{\widehat{S}}$ , defined formally by  $\text{CPre}_1(X) \equiv \{ \widehat{s} \mid \exists m_1 \in \Gamma_1(\widehat{s}) \forall m_2 \in \Gamma_2(\widehat{s}) (\widehat{\delta}_{\text{id}}(\widehat{s}, m_1, m_2) \subseteq X) \}$ . Intuitively,  $\widehat{s} \in \text{CPre}_1(X)$  iff player 1 can force the augmented game from  $\widehat{s}$  into  $X$  in one move.

### 4.3 Termination of the Fixpoint Iteration

We prove termination of the  $\mu$ -calculus fixpoint iteration by demonstrating that we can work on a finite partition of the state space. Let an equivalence relation  $\cong_e$  on the states in  $\widehat{S}$  be defined as:  $\langle\langle l^1, \kappa^1 \rangle, \mathfrak{z}^1, \beta^1, \text{tick}^1, \text{bl}_1^1 \rangle \cong_e \langle\langle l^2, \kappa^2 \rangle, \mathfrak{z}^2, \beta^2, \text{tick}^2, \text{bl}_1^2 \rangle$  iff

1.  $l^1 = l^2, \text{tick}^1 = \text{tick}^2$ , and  $\text{bl}_1^1 = \text{bl}_1^2$ .
2.  $\widehat{\kappa}^1 \cong \widehat{\kappa}^2$  where  $\widehat{\kappa}^i : C \cup \{z\} \mapsto \mathbb{R}_{\geq 0}$  is a clock valuation such that  $\widehat{\kappa}^i(c) = \kappa^i(c)$  for  $c \in C$ ,  $\widehat{\kappa}^i(z) = \mathfrak{z}^i$ , and  $c_z = 1$  ( $c_z$  is the maximum value of the clock  $z$  in the definition of  $\cong$ ) for  $i \in \{1, 2\}$ .
3.  $\beta^1 = \perp$  iff  $\beta^2 = \perp$ .
4. If  $\beta^1 \neq \perp, \beta^2 \neq \perp$  then
  - $\lfloor \beta^1 \rfloor = \lfloor \beta^2 \rfloor$
  - $\text{frac}(\beta^1) = 0$  iff  $\text{frac}(\beta^2) = 0$ .
  - For each clock  $x \in C \cup \{z\}$  with  $\kappa^1(x) \leq c_x$  and  $\kappa^2(x) \leq c_x$ , we have  $\text{frac}(\kappa^1(x)) + \text{frac}(\beta^1) \sim 1$  iff  $\text{frac}(\kappa^2(x)) + \text{frac}(\beta^2) \sim 1$  with  $\sim \in \{<, =, >\}$ .

The number of equivalence classes induced by  $\cong_e$  is again finite ( $O(|L| \cdot \prod_{x \in C} (c_x + 1) \cdot |C| + 1| \cdot 2^{|C|} \cdot |C|)$ ). We call each equivalence class an *extended region*. An extended region  $Y$  of  $\llbracket \mathcal{J} \rrbracket$  can be specified by the tuple  $\langle l, \text{tick}, \text{bl}_1, h, \mathcal{P}, \beta_i, \beta_f, C_<, C_=:, C_> \rangle$  where for a state  $\widehat{s} = \langle\langle l, \kappa \rangle, \mathfrak{z}, \beta, \text{tick}, \text{bl}_1 \rangle$ ,



**Fig. 2.** An extended region with  $C_< = C \cup \{z\}, C_ = \emptyset, C_> = \emptyset$

- $l, tick, bl_1$  correspond to  $l, tick, bl_1$  in  $\hat{s}$ .
- $h$  is a function which specifies the integer values of clocks:  $h(x) = \lfloor \kappa(x) \rfloor$  if  $\kappa(x) < C_x + 1$ , and  $h(x) = C_x + 1$  otherwise.
- $\mathcal{P} \subseteq 2^{C \cup \{z\}}$  is a partition of the clocks  $\{C_0, \dots, C_n \mid \exists C_i = C \cup \{z\}, C_i \neq \emptyset \text{ for } i > 0\}$ , such that 1) for any pair of clocks  $x, y$ , we have  $\text{frac}(\kappa(x)) < \text{frac}(\kappa(y))$  iff  $x \in C_j, y \in C_k$  for  $j < k$ ; and 2)  $x \in C_0$  iff  $\text{frac}(\kappa(x)) = 0$ .
- $\beta_i \in \mathbb{N} \cap \{0, \dots, M\} \cup \{\perp\}$  indicates the integral value of  $\beta$ .
- $\beta_f \in \{\text{TRUE}, \text{FALSE}\}$  indicates whether the fractional value of  $\beta$  is greater than 0,  $\beta_f = \text{TRUE}$  iff  $\beta \neq \perp$  and  $\text{frac}(\beta) > 0$ .
- For a clock  $x \in C \cup \{z\}$  and  $\beta \neq \perp$ , we have  $\text{frac}(\kappa(x)) + \text{frac}(\beta) \sim 1$  iff  $x \in C_{\sim}$  for  $\sim \in \{<, =, >\}$ .

Pictorially, the relationship between  $\hat{\kappa}$  and  $\beta$  can be visualised as in Fig. 2. The figure depicts an extended region for  $C_0 = \emptyset, \beta_i \in \mathbb{N} \cap \{0, \dots, M\}, \beta_f = \text{TRUE}, C_< = C \cup \{z\}, C_ = \emptyset, C_> = \emptyset$ . The vertical axis is used for the fractional value of  $\beta$ . The horizontal axis is used for the fractional values of the clocks in  $C_i$ . Thus, given a disjoint partition  $\{C_0, \dots, C_n\}$  of the clocks, we pick  $n + 1$  points on a line parallel to the horizontal axis,  $\{(C_0^f, \text{frac}(\beta)), \dots, (C_n^f, \text{frac}(\beta))\}$ , with  $C_i^f$  being the fractional value of the clocks in the set  $C_i$  at  $\hat{\kappa}$ .

**Lemma 4.** *Let  $X \subseteq \hat{S}$  consist of a union of extended regions in a timed game structure  $\llbracket \mathcal{J} \rrbracket$ . Then  $\text{CPre}_1(X)$  is again a union of extended regions.*

Lemma 4 demonstrates that the sets in the fixpoint computation of the  $\mu$ -calculus algorithm which computes winning states for player-1 for the reachability objective  $\hat{X}_p$  consist of unions of extended regions. Since the number of extended regions is finite, the algorithm terminates.

**Theorem 2.** *For a state  $s$  and a proposition  $p$  in a timed automaton game  $\mathcal{T}$ ,*

1. *The minimum time for player-1 to visit  $p$  starting from  $s$  (denoted  $T_{\min}(s, p)$ ) is computable in time  $O(|L| \cdot \prod_{x \in C} (c_x + 1) \cdot |C + 1|! \cdot 2^{|C|} \cdot |C|)$ .*
2. *For every region  $R$  of  $\llbracket \mathcal{J} \rrbracket$ , either there is a constant  $d_R \in \mathbb{N} \cup \{\infty\}$  such that for every state  $s \in R$ , we have  $T_{\min}(s, p) = d_R$ , or there is an integer constant  $d_R$  and a clock  $x \in C$  such that for every state  $s \in R$ , we have  $T_{\min}(s, p) = d_R - \text{frac}(\kappa(x))$ , where  $\kappa(x)$  is the value of the clock  $x$  in  $s$ .*

## References

1. Adler, B., de Alfaro, L., Faella, M.: Average reward timed games. In: Petterson, P., Yi, W. (eds.) FORMATS 2005. LNCS, vol. 3829, pp. 65–80. Springer, Heidelberg (2005)
2. Alur, R., Bernadsky, M., Madhusudan, P.: Optimal reachability for weighted timed games. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 122–133. Springer, Heidelberg (2004)
3. Alur, R., Dill, D.: A theory of timed automata. *Theor. Comput. Sci.* 126(2), 183–235 (1994)
4. Alur, R., Henzinger, T.: Modularity for timed and hybrid systems. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 74–88. Springer, Heidelberg (1997)
5. Asarin, E., Maler, O.: As soon as possible: Time optimal control for timed automata. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC 1999. LNCS, vol. 1569, pp. 19–30. Springer, Heidelberg (1999)
6. Bouyer, P., Cassez, F., Fleury, E., Larsen, K.G.: Optimal strategies in priced timed game automata. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 148–160. Springer, Heidelberg (2004)
7. Bouyer, P., D’Souza, D., Madhusudan, P., Petit, A.: Timed control with partial observability. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 180–192. Springer, Heidelberg (2003)
8. Brihaye, T., Bruyère, V., Raskin, J.: On optimal timed strategies. In: Petterson, P., Yi, W. (eds.) FORMATS 2005. LNCS, vol. 3829, pp. 49–64. Springer, Heidelberg (2005)
9. Cassez, F., David, A., Fleury, E., Larsen, K., Lime, D.: Efficient on-the-fly algorithms for the analysis of timed games. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 66–80. Springer, Heidelberg (2005)
10. Cassez, F., Henzinger, T., Raskin, J.-F.: A comparison of control problems for timed and hybrid systems. In: Tomlin, C.J., Greenstreet, M.R. (eds.) HSCC 2002. LNCS, vol. 2289, pp. 134–148. Springer, Heidelberg (2002)
11. Courcoubetis, C., Yannakakis, M.: Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design* 1(4), 385–415 (1992)
12. de Alfaro, L., Faella, M., Henzinger, T., Majumdar, R., Stoelinga, M.: The element of surprise in timed games. In: Amadio, R.M., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 144–158. Springer, Heidelberg (2003)
13. de Alfaro, L., Henzinger, T., Majumdar, R.: From verification to control: Dynamic programs for omega-regular objectives. In: LICS 2001, pp. 279–290. IEEE Computer Society Press, Los Alamitos (2001)
14. D’Souza, D., Madhusudan, P.: Timed control synthesis for external specifications. In: Alt, H., Ferreira, A. (eds.) STACS 2002. LNCS, vol. 2285, pp. 571–582. Springer, Heidelberg (2002)
15. Faella, M., Torre, S.L., Murano, A.: Dense real-time games. In: LICS 2002, pp. 167–176. IEEE Computer Society Press, Los Alamitos (2002)
16. Henzinger, T., Kopke, P.: Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science* 221, 369–392 (1999)
17. Henzinger, T., Prabhu, V.: Timed alternating-time temporal logic. In: Asarin, E., Bouyer, P. (eds.) FORMATS 2006. LNCS, vol. 4202, pp. 1–17. Springer, Heidelberg (2006)

18. Jurdziński, M., Trivedi, A.: Reachability-time games on timed automata. In: ICALP 2007. LNCS, Springer, Heidelberg (2007)
19. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems (an extended abstract). In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 229–242. Springer, Heidelberg (1995)
20. Pnueli, A., Asarin, E., Maler, O., Sifakis, J.: Controller synthesis for timed automata. In: Proc. System Structure and Control, Elsevier, Amsterdam (1998)
21. Segala, R., Gawlick, R., Søgaard-Andersen, J., Lynch, N.: Liveness in timed and untimed systems. *Inf. Comput.* 141(2), 119–171 (1998)
22. Wong-Toi, H., Hoffmann, G.: The control of dense real-time discrete event systems. In: Proc. of 30th Conf. Decision and Control, pp. 1527–1528 (1991)

# Reachability-Time Games on Timed Automata<sup>\*</sup>

## (Extended Abstract)

Marcin Jurdziński<sup>\*\*</sup> and Ashutosh Trivedi

Department of Computer Science, University of Warwick, UK

**Abstract.** In a reachability-time game, players Min and Max choose moves so that the time to reach a final state in a timed automaton is minimised or maximised, respectively. Asarin and Maler showed decidability of reachability-time games on strongly non-Zeno timed automata using a value iteration algorithm. This paper complements their work by providing a strategy improvement algorithm for the problem. It also generalizes their decidability result because the proposed strategy improvement algorithm solves reachability-time games on all timed automata. The exact computational complexity of solving reachability-time games is also established: the problem is EXPTIME-complete for timed automata with at least two clocks.

## 1 Introduction

Timed automata [3] are a fundamental formalism for modelling and analysis of real-time systems. They have a rich theory, mature modelling and verification tools (e.g., UPPAAL, Kronos), and have been successfully applied to numerous industrial case studies. Timed automata are finite automata augmented by a finite number of continuous real variables, which are called clocks because their values increase with time at unit rate. Every clock can be reset when a transition of the automaton is performed, and clock values can be compared to integers as a way to constrain availability of transitions. The fundamental reachability problem is PSPACE-complete for timed automata [3]. The natural optimization problems of minimizing and maximizing reachability time in timed automata are also in PSPACE [13].

The reachability (or optimal reachability-time) problems in timed automata are fundamental to the *verification* of (quantitative timing) properties of systems modelled by timed automata [3]. On the other hand, the problem of *control-program synthesis* for real-time systems can be cast as a two-player reachability (or optimal reachability-time) games, where the two players, say Min and Max, correspond to the “controller” and the “environment”, respectively, and control-program synthesis corresponds to computing winning (or optimal) strategies for Min. In other words, for control-program synthesis, we need to generalize optimization problems to *competitive optimization* problems. Reachability games [5] and reachability-time games [4] on timed automata are decidable. The former problem is EXPTIME-complete, but the elegant result of Asarin and

---

<sup>\*</sup> This research was supported in part by EPSRC project EP/E022030/1.

<sup>\*\*</sup> Part of this work was done when the author visited the Isaac Newton Institute for Mathematical Sciences, Cambridge. Financial support from the Institute is gratefully acknowledged.

Maler [4] for reachability-time games is limited to the class of strongly non-Zeno timed automata and no upper complexity bounds are given. A recent result of Henzinger and Prabhu [15] is that values of reachability-time games can be approximated for all timed automata, but computability of the exact values was left open.

**Our contribution.** We show that exact values of reachability-time games on arbitrary timed automata are uniformly computable; here uniformity means that the output of our algorithm allows us, for every starting state, to compute efficiently the value of the game starting from this state. Unlike the paper of Asarin and Maler [4], we do not require timed automata to be strongly non-Zeno. We also establish the exact complexity of reachability and reachability-time games: they are EXPTIME-complete and two clocks are sufficient for EXPTIME-hardness. For the latter result, we reduce from a recently discovered EXPTIME-complete problem of countdown games [16].

We believe that an important contribution of this paper are the novel proof techniques used. We characterize the values of the game by *optimality equations* and then we use *strategy improvement* to solve them. This allows us to obtain an elementary and constructive proof of the fundamental determinacy result for reachability-time games, which at the same time yields an efficient algorithm matching the EXPTIME lower bound for the problem. Those techniques were known for finite state systems [17,19] but we are not aware of any earlier algorithmic results based on optimality equations and strategy improvement for real-time systems such as timed automata.

**Related and future work.** A recent, concurrent, and independent work [12] establishes decidability of slightly different and more challenging reachability-time games “with the element of surprise” [14,15]. In our model of timed games, players take turns to take unilateral decisions about the duration and type of subsequent game moves. Games with surprise are more challenging in two ways: in every round of the game, players have a “time race” to be the first to perform a move; moreover, players are forbidden to use strategies which “stop the time,” because such strategies are arguably physically unrealistic and result in Zeno runs. In our reachability-time games, player Max may use such Zeno strategies in order to prevent reaching a final state. We conjecture that our techniques can be generalized to prevent player Max from using Zeno strategies.

A generalization of timed automata to priced (or weighted) timed automata [7] allows a rich variety of applications, e.g., to scheduling [6,11,18]. While the fundamental minimum reachability-price problem is PSPACE-complete [6,8], the two-player reachability-price games are undecidable on priced timed automata with at least three clocks [9]. The reachability-price games are, however, decidable for priced timed automata with one clock [11], and on the class of strongly price-non-Zeno timed automata [2,10]. Future work should include adapting the techniques of optimality equations and strategy improvement to (competitive) optimization problems on priced timed automata.

## 2 Timed Automata and Reachability-Time Games

We assume that, wherever appropriate, sets  $\mathbb{N}$  of non-negative integers and  $\mathbb{R}$  of reals contain a maximum element  $\infty$ , and we write  $\mathbb{N}_{>0}$  for the set of positive integers and  $\mathbb{R}_{\geq 0}$  for the set of non-negative reals. For  $n \in \mathbb{N}$ , we write  $\llbracket n \rrbracket_{\mathbb{N}}$  for the set  $\{0, 1, \dots, n\}$ ,



and  $\llbracket n \rrbracket_{\mathbb{R}}$  for the set  $\{r \in \mathbb{R} : 0 \leq r \leq n\}$  of non-negative reals bounded by  $n$ . For sets  $X$  and  $Y$ , we write  $[X \rightarrow Y]$  for the set of functions  $F : X \rightarrow Y$ , and  $[X \dashrightarrow Y]$  for the set of partial functions  $F : X \dashrightarrow Y$ .

**Timed automata.** Fix a constant  $k \in \mathbb{N}$  for the rest of this paper. Let  $C$  be a finite set of *clocks*. A ( $k$ -bounded) *clock valuation* is a function  $\nu : C \rightarrow \llbracket k \rrbracket_{\mathbb{R}}$ ; we write  $V$  for the set  $[C \rightarrow \llbracket k \rrbracket_{\mathbb{R}}]$  of clock valuations. If  $\nu \in V$  and  $t \in \mathbb{R}_{\geq 0}$  then we write  $\nu + t$  for the clock valuation defined by  $(\nu + t)(c) = \nu(c) + t$ , for all  $c \in C$ . For a set  $C' \subseteq C$  of clocks and a clock valuation  $\nu : C \rightarrow \mathbb{R}_{\geq 0}$ , we define  $\text{Reset}(\nu, C')(c) = 0$  if  $c \in C'$ , and  $\text{Reset}(\nu, C')(c) = \nu(c)$  if  $c \notin C'$ .

The set of *clock constraints* over the set of clocks  $C$  is the set of conjunctions of *simple clock constraints*, which are constraints of the form  $c \bowtie i$  or  $c - c' \bowtie i$ , where  $c, c' \in C$ ,  $i \in \llbracket k \rrbracket_{\mathbb{N}}$ , and  $\bowtie \in \{<, >, =, \leq, \geq\}$ . There are finitely many simple clock constraints. For every clock valuation  $\nu \in V$ , let  $\text{SCC}(\nu)$  be the set of simple clock constraints which hold in  $\nu \in V$ . A *clock region* is a maximal set  $P \subseteq V$ , such that for all  $\nu, \nu' \in P$ , we have  $\text{SCC}(\nu) = \text{SCC}(\nu')$ . In other words, every clock region is an equivalence class of the indistinguishability-by-clock-constraints relation, and vice versa. Note that  $\nu$  and  $\nu'$  are in the same clock region iff all clocks have the same integer parts in  $\nu$  and  $\nu'$ , and if the partial orders of the clocks, determined by their fractional parts in  $\nu$  and  $\nu'$ , are the same. For all  $\nu \in V$ , we write  $[\nu]$  for the clock region of  $\nu$ .

A *clock zone* is a convex set of clock valuations, which is a union of a set of clock regions. Note that a set of clock valuations is a zone iff it is definable by a clock constraint. For  $W \subseteq V$ , we write  $\overline{W}$  for the smallest closed set in  $V$  which contains  $W$ . Observe that for every clock zone  $W$ , the set  $\overline{W}$  is also a clock zone.

Let  $L$  be a finite set of *locations*. A *configuration* is a pair  $(\ell, \nu)$ , where  $\ell \in L$  is a location and  $\nu \in V$  is a clock valuation; we write  $Q$  for the set of configurations. If  $s = (\ell, \nu) \in Q$  and  $c \in C$ , then we write  $s(c)$  for  $\nu(c)$ . A *region* is a pair  $(\ell, P)$ , where  $\ell$  is a location and  $P$  is a clock region. If  $s = (\ell, \nu)$  is a configuration then we write  $[s]$  for the region  $(\ell, [\nu])$ . We write  $\mathcal{R}$  for the set of regions. A set  $Z \subseteq Q$  is a *zone* if for every  $\ell \in L$ , there is a clock zone  $W_{\ell}$  (possibly empty), such that  $Z = \{(\ell, \nu) : \ell \in L \text{ and } \nu \in W_{\ell}\}$ . For a region  $R = (\ell, P) \in \mathcal{R}$ , we write  $\overline{R}$  for the zone  $\{(\ell, \nu) : \nu \in \overline{P}\}$ .

A *timed automaton*  $\mathcal{T} = (L, C, S, A, E, \delta, \rho, F)$  consists of a finite set of locations  $L$ , a finite set of clocks  $C$ , a set of *states*  $S \subseteq Q$ , a finite set of *actions*  $A$ , an *action enabledness function*  $E : A \rightarrow 2^S$ , a *transition function*  $\delta : L \times A \rightarrow L$ , a *clock reset function*  $\rho : A \rightarrow 2^C$ , and a set of *final states*  $F \subseteq S$ . We require that  $S, F$ , and  $E(a)$  for all  $a \in A$ , are zones.

Clock zones, from which zones  $S, F$ , and  $E(a)$ , for all  $a \in A$ , are built, are typically specified by clock constraints. Therefore, when we consider a timed automaton as an input of an algorithm, its size should be understood as the sum of sizes of encodings of  $L, C, A, \delta$ , and  $\rho$ , and the sizes of encodings of clock constraints defining zones  $S, F$ , and  $E(a)$ , for all  $a \in A$ . Our definition of a timed automaton may appear to differ from the usual ones [317]. The differences are, however, superficial and mostly syntactic.

For a configuration  $s = (\ell, \nu) \in Q$  and  $t \in \mathbb{R}_{\geq 0}$ , we define  $s + t$  to be the configuration  $s' = (\ell, \nu + t)$  if  $\nu + t \in V$ , and we then write  $s \rightarrow_t s'$ . We write  $s \rightarrow_t s'$  if  $s \rightarrow_t s'$  and for all  $t' \in [0, t]$ , we have  $(\ell, s + t') \in S$ . For an action  $a \in A$ , we define  $\text{Succ}(s, a)$

to be the configuration  $s' = (\ell', \nu')$ , where  $\ell' = \delta(\ell, a)$  and  $\nu' = \text{Reset}(\nu, \rho(a))$ , and we then write  $s \xrightarrow{a} s'$ . We write  $s \xrightarrow{a} s'$  if  $s \xrightarrow{a} s'$ ;  $s, s' \in S$ ; and  $s \in E(a)$ . For technical convenience, and without loss of generality, we will assume throughout that for every  $s \in S$ , there exists  $a \in A$ , such that  $s \xrightarrow{a} s'$ .

For  $s, s' \in S$ , we say that  $s'$  is in the future of  $s$ , or equivalently, that  $s$  is in the past of  $s'$ , if there is  $t \in \mathbb{R}_{\geq 0}$ , such that  $s \rightarrow_t s'$ ; we then write  $s \rightarrow_* s'$ . For  $R, R' \in \mathcal{R}$ , we say that  $R'$  is in the future of  $R$ , or that  $R$  is in the past of  $R'$ , if for all  $s \in R$ , there is  $s' \in R'$ , such that  $s'$  is in the future of  $s$ ; we then write  $R \rightarrow_* R'$ . We say that  $R'$  is the *time successor* of  $R$  if  $R \rightarrow_* R'$ ,  $R \neq R'$ , and for every  $R'' \in \mathcal{R}$ , we have that  $R \rightarrow_* R'' \rightarrow_* R'$  implies  $R'' = R$  or  $R'' = R'$ ; we then write  $R \rightarrow_{+1} R'$  or  $R' \leftarrow_{+1} R$ . Similarly, for  $R, R' \in \mathcal{R}$ , we write  $R \xrightarrow{a} R'$  if there is  $s \in R$ , and there is  $s' \in R'$ , such that  $s \xrightarrow{a} s'$ .

We say that a region  $R \in \mathcal{R}$  is *thin* if for every  $s \in R$  and every  $\varepsilon > 0$ , we have that  $[s] \neq [s + \varepsilon]$ ; other regions are called *thick*. We write  $\mathcal{R}_{\text{Thin}}$  and  $\mathcal{R}_{\text{Thick}}$  for the sets of thin and thick regions, respectively. Note that if  $R \in \mathcal{R}_{\text{Thick}}$  then for every  $s \in R$ , there is an  $\varepsilon > 0$ , such that  $[s] = [s + \varepsilon]$ . Observe also, that the time successor of a thin region is thick, and vice versa.

A *timed action* is a pair  $\tau = (a, t) \in A \times \mathbb{R}_{\geq 0}$ . For  $s \in Q$ , we define  $\text{Succ}(s, \tau) = \text{Succ}(s, (a, t))$  to be the configuration  $s' = \text{Succ}(s + t, a)$ , i.e., such that  $s \rightarrow_t s'' \xrightarrow{a} s'$ , and we then write  $s \xrightarrow{a}_t s'$ . We write  $s \xrightarrow{a}_t s'$  if  $s \rightarrow_t s'' \xrightarrow{a} s'$ . If  $\tau = (a, t)$  then we write  $s \xrightarrow{\tau} s'$  instead of  $s \xrightarrow{a}_t s'$ , and  $s \xrightarrow{\tau} s'$  instead of  $s \xrightarrow{a}_t s'$ .

A finite run of a timed automaton is a sequence  $\langle s_0, \tau_1, s_1, \tau_2, \dots, \tau_n, s_n \rangle \in S \times ((A \times \mathbb{R}_{\geq 0}) \times S)^*$ , such that for all  $i$ ,  $1 \leq i \leq n$ , we have  $s_{i-1} \xrightarrow{\tau_i} s_i$ . For a finite run  $r = \langle s_0, \tau_1, s_1, \tau_2, \dots, \tau_n, s_n \rangle$ , we define  $\text{Length}(r) = n$ , and we define  $\text{Last}(r) = s_n$  to be the state in which the run ends. We write  $\text{Runs}_{\text{fin}}$  for the set of finite runs. An infinite run of a timed automaton is a sequence  $r = \langle s_0, \tau_1, s_1, \tau_2, \dots \rangle$ , such that for all  $i \geq 1$ , we have  $s_{i-1} \xrightarrow{\tau_i} s_i$ . For an infinite run  $r$ , we define  $\text{Length}(r) = \infty$ . For a run  $r = \langle s_0, \tau_1, s_1, \tau_2, \dots \rangle$ , we define  $\text{Stop}(r) = \inf\{i : s_i \in F\}$  and  $\text{Time}(r) = \sum_{i=1}^{\text{Length}(r)} t_i$ . We define  $\text{ReachTime}(r) = \sum_{i=1}^{\text{Stop}(r)} t_i$  if  $\text{Stop}(r) < \infty$ , and  $\text{ReachTime}(r) = \infty$  if  $\text{Stop}(r) = \infty$ , where for all  $i \geq 1$ , we have  $\tau_i = (a_i, t_i)$ .

**Strategies.** A reachability-time game  $\Gamma$  is a triple  $(\mathcal{T}, L_{\text{Min}}, L_{\text{Max}})$ , where  $\mathcal{T}$  is a timed automaton  $(L, C, S, A, E, \delta, \rho, F)$  and  $(L_{\text{Min}}, L_{\text{Max}})$  is a partition of  $L$ . We define sets  $Q_{\text{Min}} = \{(\ell, \nu) \in Q : \ell \in L_{\text{Min}}\}$ ,  $Q_{\text{Max}} = Q \setminus Q_{\text{Min}}$ ,  $S_{\text{Min}} = S \cap Q_{\text{Min}}$ ,  $S_{\text{Max}} = S \setminus S_{\text{Min}}$ ,  $\mathcal{R}_{\text{Min}} = \{[s] : s \in Q_{\text{Min}}\}$ , and  $\mathcal{R}_{\text{Max}} = \mathcal{R} \setminus \mathcal{R}_{\text{Min}}$ .

A *strategy* for Min is a function  $\mu : \text{Runs}_{\text{fin}} \rightarrow A \times \mathbb{R}_{\geq 0}$ , such that if  $\text{Last}(r) = s \in S_{\text{Min}}$  and  $\mu(r) = \tau$  then  $s \xrightarrow{\tau} s'$ . Similarly, a strategy for Max is a function  $\chi : \text{Runs}_{\text{fin}} \rightarrow A \times \mathbb{R}_{\geq 0}$ , such that if  $\text{Last}(r) = s \in S_{\text{Max}}$  and  $\chi(r) = \tau$  then  $s \xrightarrow{\tau} s'$ . We write  $\Sigma_{\text{Min}}$  and  $\Sigma_{\text{Max}}$  for the sets of strategies for Min and Max, respectively. If players Min and Max use strategies  $\mu$  and  $\chi$ , respectively, then the  $(\mu, \chi)$ -run from a state  $s$  is the unique run  $\text{Run}(s, \mu, \chi) = \langle s_0, \tau_1, s_1, \tau_2, \dots \rangle$ , such that  $s_0 = s$ , and for every  $i \geq 1$ , if  $s_i \in S_{\text{Min}}$  or  $s_i \in S_{\text{Max}}$ , then  $\mu(\text{Run}_i(s, \mu, \chi)) = \tau_{i+1}$  or  $\chi(\text{Run}_i(s, \mu, \chi)) = \tau_{i+1}$ , respectively, where  $\text{Run}_i(s, \mu, \chi) = \langle s_0, \tau_1, s_1, \dots, s_{i-1}, \tau_i, s_i \rangle$ .

We say that a strategy  $\mu$  for Min is *positional* if for all finite runs  $r, r' \in \text{Runs}_{\text{fin}}$ , we have that  $\text{Last}(r) = \text{Last}(r')$  implies  $\mu(r) = \mu(r')$ . A positional strategy for Min can be then represented as a function  $\mu : S_{\text{Min}} \rightarrow A \times \mathbb{R}_{\geq 0}$ , which uniquely determines the strategy  $\mu^\infty \in \Sigma_{\text{Min}}$  as follows:  $\mu^\infty(r) = \mu(\text{Last}(r))$ , for all finite runs  $r \in \text{Runs}_{\text{fin}}$ . Positional strategies for Max are defined and represented in the analogous way. We write  $\Pi_{\text{Min}}$  and  $\Pi_{\text{Max}}$  for the sets of positional strategies for Min and for Max, respectively.

**Value of reachability-time game.** For every  $s \in S$ , we define its *upper* and *lower values* by  $\text{Val}^*(s) = \inf_{\mu \in \Sigma_{\text{Min}}} \sup_{\chi \in \Sigma_{\text{Max}}} \text{ReachTime}(\text{Run}(s, \mu, \chi))$ , and  $\text{Val}_*(s) = \sup_{\chi \in \Sigma_{\text{Max}}} \inf_{\mu \in \Sigma_{\text{Min}}} \text{ReachTime}(\text{Run}(s, \mu, \chi))$ , respectively. The inequality  $\text{Val}_*(s) \leq \text{Val}^*(s)$  always holds.

A reachability-time game is *determined* if for every  $s \in S$ , the lower and upper values are equal to each other; then the *value*  $\text{Val}(s) = \text{Val}_*(s) = \text{Val}^*(s)$  is defined. For a strategy  $\mu \in \Sigma_{\text{Min}}$ , we define its value  $\text{Val}^\mu(s) = \sup_{\chi \in \Sigma_{\text{Max}}} \text{ReachTime}(\text{Run}(s, \mu, \chi))$ , and for  $\chi \in \Sigma_{\text{Max}}$ , we define  $\text{Val}_\chi(s) = \inf_{\mu \in \Sigma_{\text{Min}}} \text{ReachTime}(\text{Run}(s, \mu, \chi))$ . For an  $\varepsilon > 0$ , we say that a strategy  $\mu \in \Sigma_{\text{Min}}$  or  $\chi \in \Sigma_{\text{Max}}$  is  $\varepsilon$ -*optimal* if for every  $s \in S$ , we have  $\text{Val}^\mu(s) \leq \text{Val}(s) + \varepsilon$  or  $\text{Val}_\chi(s) \geq \text{Val}(s) - \varepsilon$ , respectively. Note that if a game is determined then for every  $\varepsilon > 0$ , both players have  $\varepsilon$ -optimal strategies.

We say that a reachability-time game is *positionally determined* if for every  $s \in S$ , and for every  $\varepsilon > 0$ , both players have *positional*  $\varepsilon$ -optimal strategies from  $s$ . Our results (Lemma 1, Theorem 2, and Theorem 4) yield a constructive proof of the following fundamental result for reachability-time games.

**Theorem 1 (Positional determinacy).** *Reachability-time games are positionally determined.*

**Optimality equations  $\text{Opt}_{\text{MinMax}}(\Gamma)$ .** Our principal technique is to characterize the values  $\text{Val}(s)$ , for all  $s \in S$ , as solutions of an infinite system of *optimality equations*, and then to study the equations. We write  $(T, D) \models \text{Opt}_{\text{MinMax}}(\Gamma)$ , and we say that  $(T, D)$  is a solution of *optimality equations*  $\text{Opt}_{\text{MinMax}}(\Gamma)$ , if for all  $s \in S$ , we have:

- if  $D(s) = \infty$  then  $T(s) = \infty$ ; and
- if  $s \in F$  then  $(T(s), D(s)) = (0, 0)$ ;
- if  $s \in S_{\text{Min}} \setminus F$ , then  $T(s) = \inf_{a,t} \{t + T(s') : s \xrightarrow{a}_t s'\}$ , and  $D(s) = \min \{1 + d' : T(s) = \inf_{a,t} \{t + T(s') : s \xrightarrow{a}_t s' \text{ and } D(s') = d'\}\}$ ; and
- if  $s \in S_{\text{Max}} \setminus F$ , then  $T(s) = \sup_{a,t} \{t + T(s') : s \xrightarrow{a}_t s'\}$ , and  $D(s) = \max \{1 + d' : T(s) = \sup_{a,t} \{t + T(s') : s \xrightarrow{a}_t s' \text{ and } D(s') = d'\}\}$ .

Intuitively, in the equations above,  $T(s)$  and  $D(s)$  capture “optimal time to reach a final state” and “optimal distance to reach a final state in optimal time” from state  $s \in S$ , respectively. The following key lemma establishes that in order to solve a reachability-time game  $\Gamma$ , it suffices to find a solution of  $\text{Opt}_{\text{MinMax}}(\Gamma)$ .

**Lemma 1 ( $\varepsilon$ -Optimal strategies from optimality equations).** *If  $T : S \rightarrow \mathbb{R}$  and  $D : S \rightarrow \mathbb{N}$  are such that  $(T, D) \models \text{Opt}_{\text{MinMax}}(\Gamma)$ , then for all  $s \in S$ , we have  $\text{Val}(s) = T(s)$  and for every  $\varepsilon > 0$ , both players have positional  $\varepsilon$ -optimal strategies.*

### 3 Timed Region Graph

In this section we argue that the task of solving  $\text{Opt}_{\text{MinMax}}(\Gamma)$  can be reduced to solving a simpler system of equations  $\text{Opt}_{\text{MinMax}}(\hat{\Gamma})$ , which is also infinite, but whose right-hand sides are minima or maxima of only finitely many expressions.

**Simple timed actions.** Define the finite set of *simple timed actions*  $\mathcal{A} = A \times \llbracket k \rrbracket_{\mathbb{N}} \times C$ . For  $s \in Q$  and  $\alpha = (a, b, c) \in \mathcal{A}$ , we define  $t(s, \alpha) = b - s(c)$  if  $s(c) \leq b$ , and  $t(s, \alpha) = 0$  if  $s(c) > b$ ; and we define  $\text{Succ}(s, \alpha)$  to be the state  $s' = \text{Succ}(s, \tau(\alpha))$ , where  $\tau(\alpha) = (a, t(s, \alpha))$ ; we then write  $s \xrightarrow{\alpha} s'$ . We also write  $s \xrightarrow{\alpha} s'$  if  $s \xrightarrow{\tau(\alpha)} s'$ . Note that if  $\alpha \in \mathcal{A}$  and  $s \xrightarrow{\alpha} s'$  then  $[s'] \in \mathcal{R}_{\text{Thin}}$ . Observe that for every thin region  $R' \in \mathcal{R}_{\text{Thin}}$ , there is a number  $b \in \llbracket k \rrbracket_{\mathbb{N}}$  and a clock  $c \in C$ , such that for every  $R \in \mathcal{R}$  in the past of  $R'$ , we have that  $s \in R$  implies  $(s + (b - s(c))) \in R'$ ; we then write  $R \xrightarrow{b,c} R'$ . For  $\alpha = (a, b, c) \in \mathcal{A}$  and  $R, R' \in \mathcal{R}$ , we write  $R \xrightarrow{\alpha} R'$  or  $R \xrightarrow{a,b,c} R'$ , if  $R \xrightarrow{b,c} R'' \xrightarrow{a} R'$ , for some  $R'' \in \mathcal{R}_{\text{Thin}}$ .

**Timed region graph  $\hat{\Gamma}$ .** Let  $\Gamma = (\mathcal{T}, L_{\text{Min}}, L_{\text{Max}})$  be a reachability-time game. We define the *timed region graph*  $\hat{\Gamma}$  to be the finite edge-labelled graph  $(\mathcal{R}, \mathcal{M})$ , where the set  $\mathcal{R}$  of regions of timed automaton  $\mathcal{T}$  is the set of vertices, and the labelled edge relation  $\mathcal{M} \subseteq \mathcal{R} \times \mathcal{A} \times \mathcal{R}$  is defined in the following way. For  $\alpha = (a, b, c) \in \mathcal{A}$  and  $R, R' \in \mathcal{R}$  we have  $(R, \alpha, R') \in \mathcal{M}$ , sometimes denoted by  $R \overset{\alpha}{\rightsquigarrow} R'$ , if and only if one of the following conditions holds:

- there is an  $R'' \in \mathcal{R}$ , such that  $R \xrightarrow{b,c} R'' \xrightarrow{a} R'$ ; or
- $R \in \mathcal{R}_{\text{Min}}$ , and there are  $R'', R''' \in \mathcal{R}$ , such that  $R \xrightarrow{b,c} R'' \xrightarrow{+1} R''' \xrightarrow{a} R'$ ; or
- $R \in \mathcal{R}_{\text{Max}}$ , and there are  $R'', R''' \in \mathcal{R}$ , such that  $R \xrightarrow{b,c} R'' \xrightarrow{-1} R''' \xrightarrow{a} R'$ .

Observe that in all the cases above we have that  $R'' \in \mathcal{R}_{\text{Thin}}$  and  $R''' \in \mathcal{R}_{\text{Thick}}$ . The motivation for the second case is the following. Let  $R \xrightarrow{*} R''' \xrightarrow{a} R'$ , where  $R \in \mathcal{R}_{\text{Min}}$  and  $R''' \in \mathcal{R}_{\text{Thick}}$ . One of the key results that we establish is that in a state  $s \in R$ , among all  $t \in \mathbb{R}_{\geq 0}$ , such that  $s + t \in R'''$ , the smaller the  $t$ , the “better” the timed action  $(a, t)$  is for player Min. Note, however, that the set  $\{t \in \mathbb{R}_{\geq 0} : s + t \in R'''\}$  is an open interval because  $R''' \in \mathcal{R}_{\text{Thick}}$ , and hence it does not have the smallest element. Therefore, for every  $s \in R$ , we model the “best” time to wait, when starting from  $s$ , before performing an  $a$ -labelled transition from region  $R'''$  to region  $R'$ , by taking the infimum of the set  $\{t \in \mathbb{R}_{\geq 0} : s + t \in R'''\}$ . Observe that this infimum is equal to the  $t_{R''} \in \mathbb{R}_{\geq 0}$ , such that  $s + t_{R''} \in R''$ , where  $R'' \xrightarrow{+1} R'''$ , and that  $t_{R''} = b - s(c)$ , where  $R \xrightarrow{b,c} R''$ . In the timed region graph  $\hat{\Gamma}$ , we summarize this model of the “best” timed action from region  $R$  to region  $R'$  via region  $R'''$ , by having a move  $(R, \alpha, R') \in \mathcal{M}$ , where  $\alpha = (a, b, c)$ . The motivation for the first and the third cases of the definition of  $\mathcal{M}$  is similar.

**Regional functions and optimality equations  $\text{Opt}_{\text{MinMax}}(\hat{\Gamma})$ .** Recall from Section 2 that a solution of optimality equations  $\text{Opt}_{\text{MinMax}}(\Gamma)$  for a reachability-time game  $\Gamma$  is a pair of functions  $(T, D)$ , such that  $T : S \rightarrow \mathbb{R}$  and  $D : S \rightarrow \mathbb{N}$ . Our goal is to define analogous optimality equations  $\text{Opt}_{\text{MinMax}}(\hat{\Gamma})$  for the timed region graph  $\hat{\Gamma}$ .

If  $R \overset{\alpha}{\rightsquigarrow} R'$ , where  $R, R' \in \mathcal{R}$  and  $\alpha \in \mathcal{A}$ , then  $s \in R$  does not imply that  $\text{Succ}(s, \alpha) \in R'$ ; however,  $s \in R$  implies  $\text{Succ}(s, \alpha) \in \overline{R'}$ . In order to correctly capture

the constraints for successor states which fall out of the “target” region  $R'$  of a move of the form  $R \xrightarrow{\alpha} R'$ , we consider, as solutions of optimality equations  $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$ , *regional functions* of types  $T : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$  and  $D : \mathcal{R} \rightarrow [S \rightarrow \mathbb{N}]$ , where for every  $R \in \mathcal{R}$ , the domain of partial functions  $T(R)$  and  $D(R)$  is  $\overline{R}$ . Sometimes, when defining a regional function  $F : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$ , it will only be natural to define  $F(R)$  for all  $s \in R$ , instead of all  $s \in \overline{R}$ . This is not a problem, however, because defining  $F(R)$  on the region  $R$  uniquely determines the continuous extension of  $F(R)$  to  $\overline{R}$ . For a function  $F : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$ , we define the function  $\widetilde{F} : S \rightarrow \mathbb{R}$  by  $\widetilde{F}(s) = F([s])(s)$ .

If  $F, F', G, G' : S \rightarrow \mathbb{R}$  then we write  $F \leq F'$  or  $(F, G) \leq^{\text{lex}} (F', G')$ , if for all  $s \in S$ , we have  $F(s) \leq F'(s)$  or  $(F(s), G(s)) \leq^{\text{lex}} (F'(s), G'(s))$ , respectively, where  $\leq^{\text{lex}}$  is the lexicographic order. Moreover,  $F < F'$  or  $(F, G) <^{\text{lex}} (F', G')$ , if  $F \leq F'$  or  $(F, G) \leq^{\text{lex}} (F', G')$ , and there is  $s \in S$ , such that  $F(s) < F'(s)$  or  $(F(s), G(s)) <^{\text{lex}} (F'(s), G'(s))$ , respectively.

If  $\alpha \in \mathcal{A}$ ;  $R, R' \in \mathcal{R}$ ;  $R \xrightarrow{\alpha} R'$ ; and  $T : R' \rightarrow \mathbb{R}$  and  $D : R' \rightarrow \mathbb{N}$ , then we define the functions  $T_{\alpha}^{\oplus} : R \rightarrow \mathbb{R}$  and  $D_{\alpha}^{\boxplus} : R \rightarrow \mathbb{R}$ , by  $T_{\alpha}^{\oplus}(s) = t(s, \alpha) + T(\text{Succ}(s, \alpha))$  and  $D_{\alpha}^{\boxplus}(s) = 1 + D(\text{Succ}(s, \alpha))$ , for all  $s \in R$ . We write  $(T, D) \models \text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$  if for all  $s \in S$ , we have the following:

- if  $\widetilde{D}(s) = \infty$  then  $\widetilde{T}(s) = \infty$ ; and  $(\widetilde{T}(s), \widetilde{D}(s)) = (0, 0)$  if  $s \in F$ ;
- $(\widetilde{T}(s), \widetilde{D}(s)) = \min^{\text{lex}}_{m \in \mathcal{M}} \{ (T(R')_{\alpha}^{\oplus}(s), D(R')_{\alpha}^{\boxplus}(s)) : m = ([s], \alpha, R') \}$  if  $s \in S_{\text{Min}} \setminus F$ ; and
- $(\widetilde{T}(s), \widetilde{D}(s)) = \max^{\text{lex}}_{m \in \mathcal{M}} \{ (T(R')_{\alpha}^{\oplus}(s), D(R')_{\alpha}^{\boxplus}(s)) : m = ([s], \alpha, R') \}$  if  $s \in S_{\text{Max}} \setminus F$ .

**Solutions of  $\text{Opt}_{\text{MinMax}}(\Gamma)$  from solutions of  $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$ .** In this subsection we show that the function  $(T, D) \mapsto (\widetilde{T}, \widetilde{D})$  translates solutions of  $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$  to solutions of  $\text{Opt}_{\text{MinMax}}(\Gamma)$ . In other words, the function  $\Gamma \mapsto \widehat{\Gamma}$  is a reduction from the problem of computing values in reachability-time games to the problem of solving optimality equations for timed region graphs.

In order to prove correctness of the reduction, however, we need extra properties of the solution  $(T, D)$  of  $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$ , namely that  $T$  is regionally simple, and that  $D$  is regionally constant. Let  $X \subseteq Q$ . A function  $T : X \rightarrow \mathbb{R}$  is *simple* [4] if either: there is  $e \in \mathbb{Z}$ , such that for every  $s \in X$ , we have  $T(s) = e$ ; or there are  $e \in \mathbb{Z}$  and  $c \in C$ , such that for every  $s \in X$ , we have  $T(s) = e - s(c)$ . Observe that if  $R \in \mathcal{R}$  and  $T : R \rightarrow \mathbb{R}$  is simple, then the unique continuous extension of  $T$  to  $\overline{R}$  is also simple. We say that a function  $F : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$  is *regionally simple* or *regionally constant*, if for every region  $R \in \mathcal{R}$ , the function  $F(R) : \overline{R} \rightarrow \mathbb{R}$  is simple or constant, respectively.

**Theorem 2 (Correctness of reduction to timed region graphs).** *If  $T : S \rightarrow \mathbb{R}$  and  $D : S \rightarrow \mathbb{N}$  are such that  $(T, D) \models \text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$ ,  $T$  is regionally simple, and  $D$  is regionally constant, then  $(\widetilde{T}, \widetilde{D}) \models \text{Opt}_{\text{MinMax}}(\Gamma)$ .*

## 4 Solving Optimality Equations by Strategy Improvement

In this section we give a strategy improvement algorithm to compute a solution  $(T, D)$  of  $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$ , and we argue that it satisfies the assumptions of Theorem 2, i.e.,  $T$  is regionally simple and  $D$  is regionally constant.

**Positional strategies.** A positional strategy for player Max in a timed region graph  $\widehat{T}$  is a function  $\chi : S_{\text{Max}} \rightarrow \mathcal{M}$ , such that for every  $s \in S_{\text{Max}}$ , we have  $\chi(s) = ([s], \alpha, R)$ , for some  $\alpha \in \mathcal{A}$  and  $R \in \mathcal{R}$ . A strategy  $\chi : S_{\text{Max}} \rightarrow \mathcal{M}$  is *regionally constant* if for all  $s, s' \in S_{\text{Max}}$ , we have that  $[s] = [s']$  implies  $\chi(s) = \chi(s')$ ; we can then write  $\chi([s])$  for  $\chi(s)$ . Positional strategies for player Min are defined analogously. We write  $\Delta_{\text{Max}}$  and  $\Delta_{\text{Min}}$  for the sets of positional strategies for players Max and Min, respectively.

If  $\chi \in \Delta_{\text{Max}}$  is regionally constant then we define the strategy subgraph  $\widehat{T} \upharpoonright \chi$  to be the subgraph  $(\mathcal{R}, \mathcal{M}_\chi)$  where  $\mathcal{M}_\chi \subseteq \mathcal{M}$  consists of: all moves  $(R, \alpha, R') \in \mathcal{M}$ , such that  $R \in \mathcal{R}_{\text{Min}}$ ; and of all moves  $m = (R, \alpha, R')$ , such that  $R \in \mathcal{R}_{\text{Max}}$  and  $\chi(R) = m$ . The strategy subgraph  $\widehat{T} \upharpoonright \mu$  for a regionally constant positional strategy  $\mu \in \Delta_{\text{Min}}$  for player Min is defined analogously. We say that  $R \in \mathcal{R}$  is *choiceless* in a timed region graph  $\widehat{T}$  if  $R$  has a unique successor in  $\widehat{T}$ . We say that  $\widehat{T}$  is 0-player if all  $R \in \mathcal{R}$  are choiceless in  $\widehat{T}$ ; we say that  $\widehat{T}$  is 1-player if either all  $R \in \mathcal{R}_{\text{Min}}$  or all  $R \in \mathcal{R}_{\text{Max}}$  are choiceless in  $\widehat{T}$ ; every timed region graph  $\widehat{T}$  is 2-player. Note that if  $\chi$  and  $\mu$  are positional strategies in  $\widehat{T}$  for players Max and Min, respectively, then  $\widehat{T} \upharpoonright \chi$  and  $\widehat{T} \upharpoonright \mu$  are 1-player and  $(\widehat{T} \upharpoonright \chi) \upharpoonright \mu$  is 0-player.

For functions  $T : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$  and  $D : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$ , and  $s \in S_{\text{Max}}$ , we define sets  $M^*(s, (T, D))$  and  $M_*(s, (T, D))$ , respectively, of moves enabled in  $s$  which are (lexicographically)  $(T, D)$ -optimal for player Max and Min, respectively:

$$M^*(s, (T, D)) = \operatorname{argmax}_{m \in \mathcal{M}}^{\text{lex}} \{ (T(R')_\alpha^\oplus(s), D(R')_\alpha^\boxplus(s)) : m = ([s], \alpha, R') \}, \text{ and}$$

$$M_*(s, (T, D)) = \operatorname{argmin}_{m \in \mathcal{M}}^{\text{lex}} \{ (T(R')_\alpha^\oplus(s), D(R')_\alpha^\boxplus(s)) : m = ([s], \alpha, R') \}.$$

**Optimality equations  $\text{Opt}(\widehat{T})$ ,  $\text{Opt}_{\text{Max}}(\widehat{T})$ ,  $\text{Opt}_{\text{Min}}(\widehat{T})$ ,  $\text{Opt}_{\geq}(\widehat{T})$  and  $\text{Opt}_{\leq}(\widehat{T})$ .** Let  $T : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$  and  $D : \mathcal{R} \rightarrow [S \rightarrow \mathbb{N}]$ . We write  $(T, D) \models \text{Opt}_{\text{Max}}(\widehat{T})$  or  $(T, D) \models \text{Opt}_{\text{Min}}(\widehat{T})$ , respectively, if for all  $s \in F$ , we have  $(\widetilde{T}(s), \widetilde{D}(s)) = (0, 0)$ , and for all  $s \in S \setminus F$ , we have, respectively:

$$(\widetilde{T}(s), \widetilde{D}(s)) = \max_{m \in \mathcal{M}}^{\text{lex}} \{ (T(R')_\alpha^\oplus(s), D(R')_\alpha^\boxplus(s)) : m = ([s], \alpha, R') \}, \text{ or}$$

$$(\widetilde{T}(s), \widetilde{D}(s)) = \min_{m \in \mathcal{M}}^{\text{lex}} \{ (T(R')_\alpha^\oplus(s), D(R')_\alpha^\boxplus(s)) : m = ([s], \alpha, R') \}.$$

If  $\widehat{T}$  is 0-player then  $\text{Opt}_{\text{Max}}(\widehat{T})$  and  $\text{Opt}_{\text{Min}}(\widehat{T})$  are equivalent to each other and denoted by  $\text{Opt}(\widehat{T})$ .

We write  $(T, D) \models \text{Opt}_{\geq}(\widehat{T})$  or  $(T, D) \models \text{Opt}_{\leq}(\widehat{T})$ , respectively, if for all  $s \in F$ , we have  $(\widetilde{T}(s), \widetilde{D}(s)) \geq^{\text{lex}} (0, 0)$  or  $(\widetilde{T}(s), \widetilde{D}(s)) \leq^{\text{lex}} (0, 0)$ , respectively; and for all  $s \in S \setminus F$ , we have, respectively:

$$(\widetilde{T}(s), \widetilde{D}(s)) \geq^{\text{lex}} \max_{m \in \mathcal{M}}^{\text{lex}} \{ (T(R')_\alpha^\oplus(s), D(R')_\alpha^\boxplus(s)) : m = ([s], \alpha, R') \}, \text{ or}$$

$$(\widetilde{T}(s), \widetilde{D}(s)) \leq^{\text{lex}} \min_{m \in \mathcal{M}}^{\text{lex}} \{ (T(R')_\alpha^\oplus(s), D(R')_\alpha^\boxplus(s)) : m = ([s], \alpha, R') \}.$$

The following Propositions [1](#) and [2](#) are simple but key properties of optimality equations and simple functions, which allow us to establish that if  $(T, D)$  is a solution of

0-player, 1-player, or 2-player optimality equations for  $\widehat{\Gamma}$ , then  $T$  is regionally simple and  $D$  is regionally constant.

**Proposition 1.** *Let  $\alpha \in \mathcal{A}$ ;  $R, R' \in \mathcal{R}$ ; and  $R \xrightarrow{\alpha} R'$ . If  $F : R' \rightarrow \mathbb{R}$  is simple, then  $F_\alpha^\oplus : R \rightarrow \mathbb{R}$  is simple.*

The following lemma can be proved by induction on the value of the function  $D$ , using Proposition [1](#).

**Lemma 2 (Solution of  $\text{Opt}(\widehat{\Gamma})$  is regionally simple).** *Let  $\widehat{\Gamma}$  be a 0-player timed region graph. If  $(T, D) \models \text{Opt}(\widehat{\Gamma})$  then  $T$  is regionally simple and  $D$  is regionally constant.*

### 4.1 Solving 1-Player Reachability-Time Optimality Equations $\text{Opt}_{\text{Max}}(\widehat{\Gamma})$

In this subsection we give a strategy improvement algorithm for solving maximum reachability-time optimality equations  $\text{Opt}_{\text{Max}}(\widehat{\Gamma})$  for a 1-player timed region graph  $\widehat{\Gamma}$ .

Let  $\text{Choose} : 2^{\mathcal{M}} \rightarrow \mathcal{M}$  be an arbitrary function such that for every non-empty set of moves  $M \subseteq \mathcal{M}$ , we have  $\text{Choose}(M) \in M$ . We define the following strategy improvement operator  $\text{Improve}_{\text{Max}}$ :

$$\text{Improve}_{\text{Max}}(\chi, (T, D))(s) = \begin{cases} \chi(s) & \text{if } \chi(s) \in M^*(s, (T, D)), \\ \text{Choose}(M^*(s, T)) & \text{if } \chi(s) \notin M^*(s, (T, D)). \end{cases}$$

For  $F, F' : X \rightarrow \mathbb{R}$ , we define functions  $\max(F, F'), \min(F, F') : X \rightarrow \mathbb{R}$  by  $\max(F, F')(s) = \max\{F(s), F'(s)\}$  and  $\min(F, F')(s) = \min\{F(s), F'(s)\}$ , for every  $s \in X$ . The following closure of simple functions under minimum and maximum operations, together with Proposition [1](#), yields the important Lemma [3](#).

**Proposition 2.** *Let  $F, F' : R \rightarrow \mathbb{R}$  be simple functions defined on a region  $R \in \mathcal{R}$ . Then either  $\min(F, F') = F$  and  $\max(F, F') = F'$ , or  $\min(F, F') = F'$  and  $\max(F, F') = F$ . In particular,  $\min(F, F')$  and  $\max(F, F')$  are simple functions.*

**Lemma 3 (Improvement preserves regional constancy of strategies).** *If  $\chi \in \Delta_{\text{Max}}$  is regionally constant,  $T : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$  is regionally simple, and  $D : \mathcal{R} \rightarrow [S \rightarrow \mathbb{N}]$  is regionally constant, then  $\text{Improve}_{\text{Max}}(\chi, (T, D))$  is regionally constant.*

#### Algorithm 1. Strategy improvement algorithm for $\text{Opt}_{\text{Max}}(\widehat{\Gamma})$

1. (Initialisation) Choose a regionally constant positional strategy  $\chi_0$  for player Max in  $\widehat{\Gamma}$ ; set  $i := 0$ .
2. (Value computation) Compute the solution  $(T_i, D_i)$  of  $\text{Opt}(\widehat{\Gamma} \upharpoonright \chi_i)$ .
3. (Strategy improvement) If  $\text{Improve}_{\text{Max}}(\chi_i, (T_i, D_i)) = \chi_i$ , then return  $(T_i, D_i)$ .  
Otherwise, set  $\chi_{i+1} := \text{Improve}_{\text{Max}}(\chi_i, (T_i, D_i))$ ; set  $i := i + 1$ ; and goto step 2.

**Proposition 3 (Solutions of  $\text{Opt}_{\text{Max}}(\widehat{\Gamma})$  from fixpoints of  $\text{Improve}_{\text{Max}}$ ).** *Let  $\chi \in \Delta_{\text{Max}}$  and let  $(T^\chi, D^\chi) \models \text{Opt}(\widehat{\Gamma} \upharpoonright \chi)$ . If  $\text{Improve}_{\text{Max}}(\chi, (T^\chi, D^\chi)) = \chi$  then we have  $(T^\chi, D^\chi) \models \text{Opt}_{\text{Max}}(\widehat{\Gamma})$ .*

If  $F, F', G, G' : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$  then we write  $F \leq F'$  or  $(F, G) \leq^{\text{lex}} (F', G')$ , if for all  $R \in \mathcal{R}$  and  $s \in \overline{R}$ , we have  $F(R)(s) \leq F'(R)(s)$  or  $(F(R)(s), G(R)(s)) \leq^{\text{lex}} (F'(R)(s), G'(R)(s))$ , respectively. Moreover,  $F < F'$  or  $(F, G) <^{\text{lex}} (F', G')$ , if  $F \leq F'$  or  $(F, G) \leq^{\text{lex}} (F', G')$ , and there is  $R \in \mathcal{R}$  and  $s \in R$ , such that  $F(R)(s) < F'(R)(s)$  or  $(F(R)(s), G(R)(s)) <^{\text{lex}} (F'(R)(s), G'(R)(s))$ , respectively.

The following characterization of the solution of  $\text{Opt}(\widehat{\Gamma})$  as the *maximum* solution of  $\text{Opt}_{\leq}(\widehat{\Gamma})$  yields Lemma 4 which is key for termination (see the proof of Lemma 6).

**Proposition 4 (Solution of  $\text{Opt}(\widehat{\Gamma})$  dominates solutions of  $\text{Opt}_{\leq}(\widehat{\Gamma})$ ).** *If  $(T, D) \models \text{Opt}(\widehat{\Gamma})$  and  $(T_{\leq}, D_{\leq}) \models \text{Opt}_{\leq}(\widehat{\Gamma})$ , then we have  $(T_{\leq}, D_{\leq}) \leq^{\text{lex}} (T, D)$ , and if  $(T_{\leq}, D_{\leq}) \not\models \text{Opt}(\widehat{\Gamma})$  then  $(T_{\leq}, D_{\leq}) <^{\text{lex}} (T, D)$ .*

**Lemma 4 (Strict strategy improvement for Max).** *Let  $\chi, \chi' \in \Delta_{\text{Max}}$ , let  $(T, D) \models \text{Opt}_{\text{Min}}(\widehat{\Gamma} \upharpoonright \chi)$  and  $(T', D') \models \text{Opt}_{\text{Min}}(\widehat{\Gamma} \upharpoonright \chi')$ , and let  $\chi' = \text{Improve}_{\text{Max}}(\chi, (T, D))$ . Then  $(T, D) \leq^{\text{lex}} (T', D')$  and if  $\chi \neq \chi'$  then  $(T, D) <^{\text{lex}} (T', D')$ .*

The following theorem is an immediate corollary of Lemmas 2 and 3 (the algorithm considers only regionally constant strategies), of Lemma 4 and finiteness of the number of regionally constant positional strategies for Max (the algorithm terminates), and of Proposition 3 (the algorithm returns a solution of optimality equations).

**Theorem 3 (Correctness and termination of strategy improvement).** *The strategy improvement algorithm terminates in finitely many steps and returns a solution  $(T, D)$  of  $\text{Opt}_{\text{Max}}(\widehat{\Gamma})$ , such that  $T$  is regionally simple and  $D$  is regionally constant.*

## 4.2 Solving 2-Player Reachability-Time Optimality Equations $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$

In this subsection we give a strategy improvement algorithm for solving optimality equations  $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$  for a 2-player timed region graph  $\widehat{\Gamma}$ . The structure of the algorithm is very similar to that of Algorithm 1. The only difference is that in step 2. of every iteration we solve 1-player optimality equations  $\text{Opt}_{\text{Max}}(\widehat{\Gamma} \upharpoonright \mu)$  instead of 0-player optimality equations  $\text{Opt}(\widehat{\Gamma} \upharpoonright \chi)$ . Note that we can perform step 2. of Algorithm 2 below by using Algorithm 1. Define the following strategy improvement operator  $\text{Improve}_{\text{Min}}$ :

$$\text{Improve}_{\text{Min}}(\mu, (T, D))(s) = \begin{cases} \mu(s) & \text{if } \mu(s) \in M_*(s, (T, D)), \\ \text{Choose}(M_*(s, (T, D))) & \text{if } \mu(s) \notin M_*(s, (T, D)). \end{cases}$$

**Lemma 5 (Improvement preserves regional constancy of strategies).** *If  $\mu \in \Delta_{\text{Min}}$  is regionally constant,  $T : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$  is regionally simple, and  $D : \mathcal{R} \rightarrow [S \rightarrow \mathbb{R}]$  is regionally constant, then  $\text{Improve}_{\text{Min}}(\mu, (T, D))$  is regionally constant.*

### Algorithm 2. Strategy improvement algorithm for solving $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$

1. (Initialisation) Choose a regionally constant positional strategy  $\mu_0$  for player Min in  $\widehat{\Gamma}$ ; set  $i := 0$ .
2. (Value computation) Compute the solution  $(T_i, D_i)$  of  $\text{Opt}_{\text{Max}}(\widehat{\Gamma} \upharpoonright \mu_i)$ .
3. (Strategy improvement) If  $\text{Improve}_{\text{Min}}(\mu_i, (T_i, D_i)) = \mu_i$ , then return  $(T_i, D_i)$ .  
Otherwise, set  $\mu_{i+1} := \text{Improve}_{\text{Min}}(\mu_i, (T_i, D_i))$ ; set  $i := i + 1$ ; and goto step 2.



**Proposition 5 (Fixpoints of  $\text{Improve}_{\text{Min}}$  are solutions of  $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$ ).** Let  $\mu \in \Delta_{\text{Min}}$  and  $(T^\mu, D^\mu) \models \text{Opt}_{\text{Max}}(\widehat{\Gamma} \upharpoonright \mu)$ . If  $\text{Improve}_{\text{Min}}(\mu, (T^\mu, D^\mu)) = \mu$  then we have  $(T^\mu, D^\mu) \models \text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$ .

**Proposition 6 (Solution of  $\text{Opt}_{\text{Max}}(\widehat{\Gamma})$  is dominated by solutions of  $\text{Opt}_{\geq}(\widehat{\Gamma})$ ).** If  $(T, D) \models \text{Opt}_{\text{Max}}(\widehat{\Gamma})$  and  $(T_{\geq}, D_{\geq}) \models \text{Opt}_{\geq}(\widehat{\Gamma})$ , then we have  $(T_{\geq}, D_{\geq}) \geq^{\text{lex}} (T, D)$ , and if  $(T_{\geq}, D_{\geq}) \not\models \text{Opt}_{\text{Max}}(\widehat{\Gamma})$  then  $(T_{\geq}, D_{\geq}) >^{\text{lex}} (T, D)$ .

**Lemma 6 (Strict strategy improvement for Min).** Let  $\mu, \mu' \in \Delta_{\text{Min}}$ , let  $(T, D) \models \text{Opt}_{\text{Max}}(\widehat{\Gamma} \upharpoonright \mu)$  and  $(T', D') \models \text{Opt}_{\text{Max}}(\widehat{\Gamma} \upharpoonright \mu')$ , and let  $\mu' = \text{Improve}_{\text{Min}}(\mu, (T, D))$ . Then  $(T, D) \geq^{\text{lex}} (T', D')$  and if  $\mu \neq \mu'$  then  $(T, D) >^{\text{lex}} (T', D')$ .

*Proof.* First we argue that  $(T, D) \models \text{Opt}_{\geq}(\widehat{\Gamma} \upharpoonright \mu')$  which by Proposition 6 implies that  $(T, D) \geq^{\text{lex}} (T', D')$ . Indeed for every  $s \in S \setminus F$ , if  $\mu(s) = ([s], \alpha, R)$  and  $\mu'(s) = ([s], \alpha', R')$  then we have

$$(\widetilde{T}(s), \widetilde{D}(s)) = (T(R)_{\alpha}^{\oplus}(s), D(R)_{\alpha}^{\boxplus}(s)) \geq^{\text{lex}} (T(R')_{\alpha'}^{\oplus}(s), D(R')_{\alpha'}^{\boxplus}(s)),$$

where the equality follows from  $(T, D) \models \text{Opt}_{\text{Max}}(\widehat{\Gamma} \upharpoonright \mu)$ , and the inequality follows from the definition of  $\text{Improve}_{\text{Min}}$ . Moreover, if  $\mu \neq \mu'$  then there is  $s \in S_{\text{Min}} \setminus F$  for which the above inequality is strict. Then  $(T, D) \not\models \text{Opt}_{\text{Max}}(\widehat{\Gamma} \upharpoonright \mu')$  because every vertex  $R \in \mathcal{R}_{\text{Min}}$  in  $\widehat{\Gamma} \upharpoonright \mu'$  has a unique successor, and hence again by Proposition 6 we conclude that  $(T, D) >^{\text{lex}} (T', D')$ .  $\square$

The following theorem is an immediate corollary of Theorem 3 and Lemma 5, of Lemma 6 and finiteness of the number of regionally constant positional strategies for Min, and of Proposition 5.

**Theorem 4 (Correctness and termination of strategy improvement).** The strategy improvement algorithm terminates in finitely many steps and returns a solution  $(T, D)$  of  $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma})$ , such that  $T$  is regionally simple and  $D$  is regionally constant.

## 5 Complexity

**Lemma 7 (Complexity of strategy improvement).** Let  $\widehat{\Gamma}_0$ ,  $\widehat{\Gamma}_1$ , and  $\widehat{\Gamma}_2$  be 0-player, 1-player, and 2-player timed region graphs, respectively. A solution of  $\text{Opt}(\widehat{\Gamma}_0)$  can be computed in time  $O(|\mathcal{R}|)$ . The strategy improvement algorithms for  $\text{Opt}_{\text{Max}}(\widehat{\Gamma}_1)$  and  $\text{Opt}_{\text{MinMax}}(\widehat{\Gamma}_2)$  terminate in  $O(|\mathcal{R}|)$  iterations.

Since the number  $|\mathcal{R}|$  of regions is at most exponential in the size of a timed automaton [3], it follows that the strategy improvement algorithm runs in exponential time, and hence solving reachability-time games is in EXPTIME. The reachability problem for timed automata with three clocks is PSPACE-complete [13]. We show that solving 2-player reachability games on timed automata with two clocks is EXPTIME-complete. We use a reduction from countdown games [16] for EXPTIME-hardness.

**Theorem 5 (Complexity of reachability(-time) games on timed automata).** Problems of solving reachability and reachability-time games are EXPTIME-complete on timed automata with at least two clocks.

## References

1. Abdeddaïm, Y., Asarin, E., Maler, O.: Scheduling with timed automata. *Theor. Comput. Sci.* 354(2), 272–300 (2006)
2. Alur, R., Bernadsky, M., Madhusudan, P.: Optimal reachability for weighted timed games. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) *ICALP 2004*. LNCS, vol. 3142, pp. 122–133. Springer, Heidelberg (2004)
3. Alur, R., Dill, D.L.: A theory of timed automata. *Theor. Comput. Sci.* 126(2), 183–235 (1994)
4. Asarin, E., Maler, O.: As soon as possible: Time optimal control for timed automata. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) *HSCC 1999*. LNCS, vol. 1569, pp. 19–30. Springer, Heidelberg (1999)
5. Asarin, E., Maler, O., Pnueli, A., Sifakis, J.: Controller synthesis for timed automata. In: *IFAC Symp. on System Structure and Control*, pp. 469–474. Elsevier, Amsterdam (1998)
6. Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Pettersson, P., Romijn, J., Vaandrager, F.W.: Minimum-cost reachability for priced timed automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) *HSCC 2001*. LNCS, vol. 2034, pp. 147–161. Springer, Heidelberg (2001)
7. Bouyer, P.: Weighted timed automata: model-checking and games. *Electr. Notes Theor. Comput. Sci.* 158, 3–17 (2006)
8. Bouyer, P., Brihaye, T., Bruyère, V., Raskin, J.-F.: On the optimal reachability problem on weighted timed automata. *Form. Method. Syst. Des.* (to appear)
9. Bouyer, P., Brihaye, T., Markey, N.: Improved undecidability results on weighted timed automata. *Information Processing Letters* 98(5), 188–194 (2006)
10. Bouyer, P., Cassez, F., Fleury, E., Larsen, K.G.: Optimal strategies in priced timed game automata. In: Lodaya, K., Mahajan, M. (eds.) *FSTTCS 2004*. LNCS, vol. 3328, pp. 148–160. Springer, Heidelberg (2004)
11. Bouyer, P., Larsen, K.G., Markey, N., Rasmussen, J.I.: Almost optimal strategies in one-clock priced timed automata. In: Arun-Kumar, S., Garg, N. (eds.) *FSTTCS 2006*. LNCS, vol. 4337, pp. 345–356. Springer, Heidelberg (2006)
12. Brihaye, T., Henzinger, T.A., Prabhu, V.S., Raskin, J.-F.: Minimum-time reachability in timed games. In: *ICALP 2007*. LNCS, vol. 4596, Springer, Heidelberg (2007)
13. Courcoubetis, C., Yannakakis, M.: Minimum and maximum delay problems in real-time systems. *Form. Method. Syst. Des.* 1, 385–415 (1992)
14. de Alfaro, L., Faella, M., Henzinger, T.A., Majumdar, R., Stoelinga, M.: The element of surprise in timed games. In: Amadio, R.M., Lugiez, D. (eds.) *CONCUR 2003*. LNCS, vol. 2761, pp. 144–158. Springer, Heidelberg (2003)
15. Henzinger, T.A., Prabhu, V.S.: Timed alternating-time temporal logic. In: Asarin, E., Bouyer, P. (eds.) *FORMATS 2006*. LNCS, vol. 4202, pp. 1–17. Springer, Heidelberg (2006)
16. Jurdziński, M., Laroussinie, F., Sproston, J.: Model checking probabilistic timed automata with one or two clocks. In: *TACAS 2007*. LNCS, vol. 4424, pp. 170–184. Springer, Heidelberg (2007)
17. Puterman, M.L.: *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, Chichester (1994)
18. Rasmussen, J.I., Larsen, K.G., Subramani, K.: On using priced timed automata to achieve optimal scheduling. *Form. Method. Syst. Des.* 29, 97–114 (2006)
19. Vöge, J., Jurdziński, M.: A discrete strategy improvement algorithm for solving parity games. In: Emerson, E.A., Sistla, A.P. (eds.) *CAV 2000*. LNCS, vol. 1855, pp. 202–215. Springer, Heidelberg (2000)

# Perfect Information Stochastic Priority Games

Hugo Gimbert<sup>1</sup> and Wiesław Zielonka<sup>2,\*</sup>

<sup>1</sup> LIX, École Polytechnique, Palaiseau, France  
gimbert@lix.polytechnique.fr

<sup>2</sup> LIAFA, Université Paris 7 and CNRS, Paris, France  
zielonka@liafa.jussieu.fr

**Abstract.** We introduce stochastic priority games — a new class of perfect information stochastic games. These games can take two different, but equivalent, forms. In stopping priority games a play can be stopped by the environment after a finite number of stages, however, infinite plays are also possible. In discounted priority games only infinite plays are possible and the payoff is a linear combination of the classical discount payoff and of a limit payoff evaluating the performance at infinity. Shapley games [1] and parity games [2] are special extreme cases of priority games.

## 1 Introduction

Recently de Alfaro, Henzinger and Majumdar [3] introduced a new variant of  $\mu$ -calculus: discounted  $\mu$ -calculus. As it is known since the seminal paper [2] of Emerson and Jutla  $\mu$ -calculus is strongly related to parity games and this relationship is preserved even for stochastic games, [4]. In this context it is natural to ask if there is a class of games that corresponds to discounted  $\mu$ -calculus of [3]. A partial answer to this question was given in [5], where an appropriate class of infinite discounted games was introduced. However, in [5], only deterministic systems were considered and much more challenging problem of stochastic games was left open. In the present paper we return to the problem but in the context of perfect information stochastic games. The most basic and usually non-trivial question is if the games that we consider admit “simple” optimal strategies for both players. We give a positive answer, for all games presented in this paper both players have pure stationary optimal strategies. Since our games contain parity games as a very special case, our paper extends the result known for perfect information parity games [6,7,8,9].

However, we have an objective which is larger than just transferring to stochastic games the results known for deterministic systems. Parity games are used (directly or through an associated logic) in verification. Conditions that are verified often do not depend on any finite prefix of the play (take as a typical example a simple condition like “A wins if we visit infinitely often some set  $X$  of states”). However, certainly all real systems have a finite life span thus we can ask what

---

\* Partially supported by the french ANR-SETI project AVERISS.

is the meaning of infinite games when they are used to examine such systems. Notice that the same question arises in classical game theory [10]. The obvious answer is that the life span is finite but unknown or sufficiently long and thus infinite games are a convenient approximation of finite games. However, what finite games are approximated by parity games? Notice that for the games like mean-payoff games that are used in economics the answer is simple: infinite mean-payoff games approximate finite mean-payoff games of long or unknown duration. But we do not see any obvious candidate for “finite parity games”. Suppose that  $C$  is a parity condition and  $f_C$  a payoff mapping associated with  $C$ , i.e.  $f_C$  maps to 1 (win) all infinite sequence of states that satisfy  $C$  and to 0 all “loosing” sequences. Now we can look for a sequence  $f_n, n \in \mathbb{N}$ , of payoff functions, such that each  $f_n$ , defined for state sequences of length  $n$ , gives a payoff for games of length  $n$  and such that for each infinite sequence  $s_0s_1\dots$  of states  $f_n(s_0\dots s_{n-1}) \xrightarrow[n \rightarrow \infty]{} f_C(s_0s_1\dots)$ . However, except for very special parity conditions  $C$ , such payoff mappings  $f_n$  do not exist, thus parity games cannot approximate finite games in the same way as infinite mean-payoff games approximate finite mean-payoff games.

Nevertheless, it turns out that parity games approximate finite games, however “finite” does not mean here that the number of steps is fixed, instead these games are finite in the sense that they stop with probability 1. In Section 4 we present a class of priority stopping games. In the simplest case, when the stopping probabilities are positive for all states, stopping games are stochastic games defined by Shapley [1]. However, we examine also stopping games for which stopping probabilities are positive only for some states. One of the results of this paper can be interpreted in the following way: parity games are a limit of stopping games when the stopping probabilities tend to 0 but all at the same time but rather one after another, in the order determined by priorities.

## 2 Arenas and Perfect Information Games

Perfect information stochastic games are played by two players, that we call player 1 and player 2. We assume that player  $i \in \{1, 2\}$  controls a finite set  $S_i$  of states,  $S_1$  and  $S_2$  are disjoint and  $S = S_1 \cup S_2$  is the set of all states.

With each state  $s \in S$  is associated a finite non-empty set  $A_s$  of actions that are available at  $s$  and we set  $A = \cup_{s \in S} A_s$  to be the set of all actions.

If the current state is  $s \in S_i$  then player  $i$  controlling this state chooses an available action  $a \in A_s$  and, with a probability  $p(s'|s, a)$ , the systems changes its state to  $s' \in S$ . Thus  $p(\cdot|s, a), s \in S, a \in A_s$ , are transition probabilities satisfying the usual conditions:  $0 \leq p(s'|s, a) \leq 1$  and  $\sum_{s' \in S} p(s'|s, a) = 1$ .

Let  $\mathcal{H}^\omega$  be the set of *histories*, i.e. the set of all infinite sequences  $s_0a_0s_1a_1s_2\dots$  alternating states and actions. Assuming that the sets  $S$  and  $A$  are equipped with the discrete topology, we equip  $\mathcal{H}^\omega$  with the product topology, i.e. the smallest topology for which the mappings

$$\mathbf{S}_i : \mathcal{H}^\omega \rightarrow S, \quad \mathbf{S}_i : \mathcal{H}^\omega \ni s_0a_0\dots s_ia_i\dots \mapsto s_i$$

and

$$\mathbf{A}_i : \mathcal{H}^\omega \rightarrow A, \quad \mathbf{A}_i : \mathcal{H}^\omega \ni s_0 a_0 \dots s_i a_i \dots \mapsto a_i$$

are continuous. Thus  $(\mathbf{S}_i)_{i \in \mathbb{N}}$  and  $(\mathbf{A}_i)_{i \in \mathbb{N}}$ , are stochastic processes on the probability space  $(\mathcal{H}^\omega, \mathcal{B})$ , where  $\mathcal{B}$  is Borel  $\sigma$ -algebra generated by open subsets of  $\mathcal{H}^\omega$ .

The data consisting of the state sets  $S_1, S_2$ , available actions  $(A_s)_{s \in S}$  and transition probabilities  $p(\cdot, s, a)$  is an arena  $\mathcal{A}$ .

Let  $u : \mathcal{H}^\omega \rightarrow \mathbb{R}$  be a bounded Borel measurable mapping. We interpret  $u(h), h \in \mathcal{H}^\omega$ , as the payoff obtained by player 1 from player 2 after an infinite play  $h$ .

A couple  $(\mathcal{A}, u)$  consisting of an arena and a payoff mapping is a *perfect information stochastic game*.

Let  $\mathcal{H}_i^+ = (SA)^* S_i, i \in \{1, 2\}$ , be the set of finite non-empty histories terminating at a state controlled by player  $i$ . A *strategy* for player  $i$  is a family of conditional probabilities  $\sigma(a|h_n)$  for all  $h_n = s_0 a_0 \dots s_n \in \mathcal{H}_i^+$  and  $a \in A_{s_n}$ . Intuitively,  $\sigma(a|s_0 a_0 \dots s_n)$  gives the probability that player  $i$  controlling the last state  $s_n$  chooses an (available) action  $a$ , while the sequence  $h_n$  describes the first  $n$  steps of the game. As usual  $0 \leq \sigma(a|s_0 a_0 \dots s_n) \leq 1$  and  $\sum_{a \in A_{s_n}} \sigma(a|s_0 a_0 \dots s_n) = 1$ .

A strategy  $\sigma$  is said to be *pure* if for each finite history  $h_n = s_0 a_0 \dots s_n \in \mathcal{H}_1^+$  there is an action  $a \in A_{s_n}$  such that  $\sigma(a|h_n) = 1$ , i.e. no randomization is used to choose an action to execute. A strategy  $\sigma$  is *stationary* if for each finite history  $h_n = s_0 a_0 \dots s_n \in \mathcal{H}_1^+, \sigma(\cdot|h_n) = \sigma(\cdot|s_n)$ , i.e. the probability distribution used to choose actions depends only on the last state.

Notice that pure stationary strategies for player  $i$  can be identified with mappings  $\sigma : S_i \rightarrow A$  such that  $\sigma(s) \in A_s$  for  $s \in S_i$ .

In the sequel we shall use  $\sigma$ , possibly with subscripts or superscripts, to denote a strategy of player 1. On the other hand,  $\tau$  will always denote a strategy of player 2.

Given an initial state  $s$ , strategies  $\sigma, \tau$  of both players determine a unique probability measure  $\mathbb{P}_{\sigma, \tau}^s$  on  $(\mathcal{H}^\omega, \mathcal{B})$ , [11].

The expectation corresponding to the probability measure  $\mathbb{P}_{\sigma, \tau}^s$  is denoted  $\mathbb{E}_{\sigma, \tau}^s$ . Thus  $\mathbb{E}_{\sigma, \tau}^s(u)$  gives the expected payoff obtained by player 1 from player 2 in the game  $(\mathcal{A}, u)$  starting at state  $s$  when the players use strategies  $\sigma, \tau$  respectively. If  $\sup_\sigma \inf_\tau \mathbb{E}_{\sigma, \tau}^s(u) = \inf_\tau \sup_\sigma \mathbb{E}_{\sigma, \tau}^s(u)$  for each state  $s$  then the quantity appearing on both side of this equality is *the value of the game* (for initial state  $s$ ) and is denoted  $\text{val}_s(\mathcal{A}, u)$ .

Strategies  $\sigma^\sharp$  and  $\tau^\sharp$  of players 1, 2 are *optimal* in the game  $(\mathcal{A}, u)$  if for each state  $s \in S$  and for all strategies  $\sigma \in \Sigma, \tau \in \mathcal{T}$

$$\mathbb{E}_{\sigma, \tau^\sharp}^s[u] \leq \mathbb{E}_{\sigma^\sharp, \tau^\sharp}^s[u] \leq \mathbb{E}_{\sigma^\sharp, \tau}^s[u] .$$

If  $\sigma^\sharp$  and  $\tau^\sharp$  are optimal strategies then  $\text{val}_s(\mathcal{A}, u) = \mathbb{E}_{\sigma^\sharp, \tau^\sharp}^s[u]$ , i.e. the expected payoff obtained when both players use optimal strategies is equal to the value of the game.

### 3 Priority Games

Starting from this moment we assume that each arena  $\mathcal{A}$  is equipped with a *priority mapping*

$$\varphi : S \rightarrow \{1, \dots, k\} \tag{1}$$

from the set  $S$  of states to the set  $\{1, \dots, k\}$  of (positive integer) *priorities*. The composition

$$\varphi_n = \varphi \circ \mathbf{S}_n, \quad , n \in \mathbb{N} , \tag{2}$$

$\varphi_n : \mathcal{H}^\omega \rightarrow \{1, \dots, k\}$ , gives therefore a stochastic process with values in  $\{1, \dots, k\}$ . Then  $\liminf_i \varphi_i$  is a random variable

$$\mathcal{H}^\omega \ni h \mapsto \liminf_i \varphi_i(h)$$

giving for each infinite history  $h \in \mathcal{H}^\omega$  its priority which the smallest priority visited infinitely often in  $h$  (we assume that  $\{1, \dots, k\}$  is equipped with the usual order on integers and  $\liminf$  is taken for this order). From this moment onward, we assume that there is a fixed a *reward mapping*

$$r : \{1, \dots, k\} \rightarrow [0, 1] \tag{3}$$

from priorities to the interval  $[0, 1]$ .

The *priority payoff mapping*  $u : \mathcal{H}^\omega \rightarrow [0, 1]$  is defined as

$$u(h) = r(\liminf_i \varphi_i(h)), \quad h \in \mathcal{H}^\omega . \tag{4}$$

Thus, in the priority game  $(\mathcal{A}, u)$ , the payoff received by player 1 from player 2 is the reward corresponding to the minimal priority visited infinitely often. If  $r$  maps odd priorities to 1 and even priorities to 0 then we get a parity game.

One of the referees drew our attention to the paper of McIver and Morgan [12] where a new stochastic game is introduced and it is proved that this game admits pure stationary optimal strategies. The framework used in [12] is so different from the one used in our paper that the direct comparison is difficult. However, it seems that, after an appropriate translation, the game of [12] corresponds to the priority game defined above thus [12] gives a direct proof that priority games admit pure stationary optimal strategies through a reduction to parity games. In our paper we define, in Sections 4 and 5, a much larger class of games, that includes priority games as a special case and not only we prove that all these games admit pure stationary optimal strategies but we show also that all these games can be seen as appropriate limits of classical discounted games of Shapley [1].

### 4 Stopping Priority Games

In the sequel we assume that besides the priority and reward mappings (1) and (3) we have also a mapping

$$\lambda : \{1, \dots, k\} \rightarrow [0, 1] \tag{5}$$

from priorities to the interval  $[0, 1]$ .

We modify the rules of the priority game of Section 3 in the following way.

Every time a state  $s$  is visited the game can stop with probability  $1 - \lambda(\varphi(s))$ , where  $\varphi(s)$  is the priority of  $s$ . If the game stops at  $s$  then player 1 receives from player 2 the payoff  $r(\varphi(s))$ . If the game does not stop then the player controlling  $s$  chooses an action  $a \in A_s$  and we go to a state  $t$  with probability  $p(t|s, a)$ . (Thus  $p(t|s, a)$  should now be interpreted as the probability to go to  $t$  under the condition that the games does not stop.) The rules above determine the payoff in the case when the games stops at some state  $s$ . However,  $\lambda$  can be 1 for some states (priorities) and then it is possible to have also infinite plays with a positive probability. For such infinite plays the payoff is calculated as in priority games of the preceding section.

Let us note that if  $\lambda(p) = 1$  for all priorities  $p \in \{1, \dots, k\}$  then actually we never stop and the game described above is the same as the priority game of the preceding section.

On the other hand, if  $\lambda(p) < 1$  for all priorities  $p$ , i.e. the stopping probabilities are positive for all states, then the game will stop with probability 1. Shapley [11] proved that for such games both players have optimal stationary strategies. In fact Shapley considered general stochastic games while we limit ourselves to perfect information stochastic games and for such games the optimal strategies constructed in [11] are not only stationary but also pure.

**Theorem 1 (Shapley 1953).** *If, for all priorities  $i$ ,  $\lambda(i) < 1$  then both players have pure stationary optimal strategies in the priority stopping game.*

Stopping games have an appealing intuitive interpretation but they are not consistent with the framework fixed in Section 2, where the probability space consisted of infinite histories only. This obstacle can be removed in the following way. For each priority  $i \in \{1, \dots, k\}$  we create a new “stopping” state  $i^\sharp$  that we add to the arena  $\mathcal{A}$ . The priority of  $i^\sharp$  is set to  $i$ ,  $\varphi(i^\sharp) = i$ . The set of newly created states is denoted  $S^\sharp$ . There is only one action available at each  $i^\sharp$  and executing this action we return immediately to  $i^\sharp$  with probability 1, it is impossible to leave a stopping state. Note also that since there is only one action available at  $i^\sharp$  it does not matter which of the two players controls “stopping” states. For each non-stopping state  $s \in S$  we modify the transition probabilities. Formally we define new transition probabilities  $p^\sharp(\cdot|\cdot, \cdot)$  by setting, for  $s, t \in S$ ,  $a \in A_s$ ,

$$p^\sharp(t|s, a) = \lambda(\varphi(s)) \cdot p(t|s, a)$$

and

$$p^\sharp(i^\sharp|s, a) = \begin{cases} 1 - \lambda(\varphi(s)) & \text{if } i = \varphi(s), \\ 0 & \text{otherwise} . \end{cases}$$

Let us note by  $\mathcal{A}_\lambda^\sharp$  the arena obtained from  $\mathcal{A}$  in this way. It is worth noticing that, even if the set of finite histories of  $\mathcal{A}_\lambda^\sharp$  strictly contains the set of finite histories of  $\mathcal{A}$ , we can identify the strategies in both arenas. In fact, given a strategy for arena  $\mathcal{A}$  there is only one possible way to complete it to a strategy in  $\mathcal{A}_\lambda^\sharp$  since for finite histories in  $\mathcal{A}_\lambda^\sharp$  that end in a stopping state  $i^\sharp$  any strategy

chooses always the unique action available at  $i^\sharp$ . Clearly, playing a stopping priority game on  $\mathcal{A}$  is the same as playing priority game on  $\mathcal{A}_\lambda^\sharp$ : stopping at state  $s$  in  $\mathcal{A}$  yields the same payoff as an infinite history in  $\mathcal{A}_\lambda^\sharp$  that loops at  $i^\sharp$ , where  $i = \varphi(s)$ .

### 5 Discounted Priority Games

The aim of this section is to introduce a new class of infinite games that are equivalent to stopping priority games.

As previously, we suppose that arenas are equipped with a priority mapping (1) and that a reward mapping (3) is fixed.

On the other hand, the mapping  $\lambda$  of (5), although also present, has now another interpretation, it does not define stopping probabilities but it provides discount factors applied to one-step rewards.

Let

$$r_i = r \circ \varphi_i \quad \text{and} \quad \lambda_i = \lambda \circ \varphi_i, \quad i \in \mathbb{N} , \tag{6}$$

be stochastic processes giving respectively the reward and the discount factor at stage  $i$ . Then the payoff mapping  $u_\lambda : \mathcal{H}^\omega \rightarrow \mathbb{R}$  of *discounted priority games* is defined as

$$u_\lambda = \sum_{i=0}^{\infty} \lambda_0 \cdots \lambda_{i-1} (1 - \lambda_i) r_i + \left( \prod_{i=0}^{\infty} \lambda_i \right) \cdot r(\liminf_n \varphi_n) . \tag{7}$$

Thus  $u_\lambda$  is composed of two parts, the *discount part*

$$u_\lambda^{\text{disc}} = \sum_{i=0}^{\infty} \lambda_0 \cdots \lambda_{i-1} (1 - \lambda_i) r_i \tag{8}$$

and the *limit part*

$$u_\lambda^{\text{lim}} = \left( \prod_{i=0}^{\infty} \lambda_i \right) \cdot r(\liminf_n \varphi_n) . \tag{9}$$

Some remarks concerning this definition are in order. Let

$$T = \inf\{i \mid \lambda_j = 1 \text{ for all } j \geq i\} . \tag{10}$$

Since, by convention, the infimum of the empty set is  $\infty$ ,  $\{T = \infty\}$  consists of all infinite histories  $h \in \mathcal{H}^\omega$  for which  $\lambda_i < 1$  for infinitely many  $i$ . Thus we can rewrite  $u_\lambda$  as:

$$u_\lambda = \sum_{i < T} \lambda_0 \cdots \lambda_{i-1} (1 - \lambda_i) r_i + \left( \prod_{i < T} \lambda_i \right) \cdot r(\liminf_n \varphi_n) . \tag{11}$$

Moreover, if  $T = \infty$  then the product  $\prod_{i < T} \lambda_i$ , containing infinitely many factors smaller than 1, is equal to 0 and for such infinite histories the limit part  $u_\lambda^{\text{lim}}$  disappears while the discount part is (a sum of) an infinite series. The other



extreme case is  $T = 0$ , i.e. when the discount factor is 1 for all visited states. Then it is the the discount part that disappears from [11](#) and the payoff is just  $r(\liminf_n \varphi_n)$ , the same as for priority games of [Section 3](#).

Let  $(\mathcal{A}, u_\lambda)$  be a discounted priority game on  $\mathcal{A}$ . As explained in the preceding section, a stopping priority game on  $\mathcal{A}$  with stopping probabilities given by means of  $\lambda$  can be identified with the priority game  $(\mathcal{A}_\lambda^\sharp, u)$  on the transformed arena  $\mathcal{A}_\lambda^\sharp$ . As noted also in the preceding section, there is a natural correspondence allowing to identify strategies in both arenas. We shall note by  $\mathbb{P}_{\sigma,\tau}^{\sharp s}$  the probability generated by strategies  $\sigma$  and  $\tau$  on  $\mathcal{A}_\lambda^\sharp$  and  $\mathbb{P}_{\sigma,\tau}^s$  the similar probability generated by the same strategies on  $\mathcal{A}$ . The corresponding expectations are denoted  $\mathbb{E}_{\sigma,\tau}^{\sharp s}$  and  $\mathbb{E}_{\sigma,\tau}^s$ . Having all this facts in mind, the following proposition shows that stopping priority games and discounted priority games are equivalent in the sense that the same strategies yield the same payoffs in both games:

**Proposition 1.** *For all strategies  $\sigma, \tau$  of players 1, 2 and all states  $s \in S$ ,  $\mathbb{E}_{\sigma,\tau}^{\sharp s}[u] = \mathbb{E}_{\sigma,\tau}^s[u_\lambda]$ .*

*Proof.* (sketch) Let  $T = \inf\{i \mid \mathbf{S}_i \in S^\sharp\}$  be the first moment in the game  $(\mathcal{A}_\lambda^\sharp, u)$  when we enter a stopping state. Direct calculations show that  $\mathbb{P}_{\sigma,\tau}^{\sharp s}(\mathbf{S}_{i+1} = s_{i+1} \mid \mathbf{S}_0 = s_0, \dots, \mathbf{S}_i = s_i) = \lambda(\varphi(s_i))\mathbb{P}_{\sigma,\tau}^s(\mathbf{S}_{i+1} = s_{i+1} \mid \mathbf{S}_0 = s_0, \dots, \mathbf{S}_i = s_i)$  if all states  $s_0, \dots, s_i, s_{i+1}$  are not stopping. This can be used to show that [11](#)  $\mathbb{E}_{\sigma,\tau}^{\sharp s}[u; T = \infty] = \mathbb{E}_{\sigma,\tau}^s[u_\lambda^{\text{lim}}]$ . On the other hand,  $\mathbb{E}_{\sigma,\tau}^{\sharp s}[u; T = m] = \mathbb{E}_{\sigma,\tau}^s[\lambda_0 \cdots \lambda_{m-1}(1 - \lambda_m)\mathbf{r}_m]$ , implying  $\mathbb{E}_{\sigma,\tau}^{\sharp s}[u; T < \infty] = \mathbb{E}_{\sigma,\tau}^s[u_\lambda^{\text{disc}}]$ .  $\square$

We can note that in the special case when all discount factors are strictly smaller than 1 (i.e. all stopping probabilities are greater than 0) [Proposition 1](#) reduces to a well-known folklore fact: stopping (Shapley) games [11](#) and discounted games are equivalent.

## 6 Limits of Priority Discounted Games

The main aim of this section is to prove that discounted priority games  $(\mathcal{A}, u_\lambda)$  admit pure stationary optimal strategies for both players. Of course, due to Shapley’s theorem, we already know that this is true for discounted mappings  $\lambda$  such that  $\lambda(i) < 1$  for all priorities  $i$ . Our proof will use in an essential way the concept of uniformly optimal strategies, which is of independent interest.

Let  $\lambda_1, \dots, \lambda_m, 1 \leq m \leq k$ , be a sequence of constants, all belonging to the right-open interval  $[0, 1)$ . Let  $\lambda$  be the following discount mapping:

$$\text{for all } i \in \{1, \dots, k\}, \quad \lambda(i) = \begin{cases} \lambda_i & \text{if } i \leq m, \\ 1 & \text{if } i > m. \end{cases} \tag{12}$$

In the sequel we shall write  $u_{\lambda_1, \dots, \lambda_m}^{(k)}$  to denote the discounted priority payoff mapping  $u_\lambda$ , where  $\lambda$  is given by [\(12\)](#). (Note, however, that one should not

<sup>1</sup> By  $\mathbb{E}_{\sigma,\tau}^{\sharp s}[u; A]$  we denote the integral of  $u$  over the set  $A$ .

confuse  $\lambda_1, \lambda_2, \dots$  which are used to denote real numbers from  $[0, 1]$  with bold  $\lambda_1, \lambda_2, \dots$  that are used to denote a stochastic process (6).

Defining  $u_{\lambda_1, \dots, \lambda_m}^{(k)}$  we have assumed that  $m \geq 1$ , however it is convenient to include the case  $m = 0$  in this notation: if  $m = 0$  then  $\lambda(i) = 1$  for all priorities  $i \in \{1, \dots, k\}$  and thus  $u^{(k)}$  is just the priority payoff mapping  $u$  of Section 3. In particular if  $m = 1$  then  $u_{\lambda_1, \dots, \lambda_{m-1}}^{(k)}$  denotes just  $u^{(k)} = u$ .

Which strategies are optimal in the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_m}^{(k)})$  usually depends heavily on the discount factors  $\lambda_1, \dots, \lambda_m$ . But, in an important paper [13] Blackwell observed that in discounted Markov decision processes optimal strategies are independent of the discount factor if this factor is close to 1. This leads to the concept of uniformly optimal strategies:

**Definition 1.** Let  $\mathcal{A}$  be a finite arena,  $m \in \{1, \dots, k\}$ . Let us fix values of the first  $m - 1$  discount factors  $\lambda_1, \dots, \lambda_{m-1} \in [0, 1]$ . Strategies  $\sigma, \tau$  for players 1, 2 are said to be uniformly optimal for  $\lambda_1, \dots, \lambda_{m-1}$  if there exists an  $\epsilon > 0$  (that can depend on  $\lambda_1, \dots, \lambda_{m-1}$ ) such that  $\sigma, \tau$  are optimal for all games  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)})$  with  $1 - \epsilon < \lambda_m < 1$ .

Now we are prepared to announce the main result of the paper:

**Theorem 2.** For each  $m \in \{1, \dots, k\}$  the games  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)})$  admit pure stationary uniformly optimal strategies for both players. Moreover, if  $(\sigma^\#, \tau^\#)$  is a pair of such strategies then  $\sigma^\#, \tau^\#$  are also optimal in the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}}^{(k)})$ .

Proposition 2 below establishes the following chain of implications:

if  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_m}^{(k)})$  admits pure stationary optimal strategies then  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_m}^{(k)})$  admits pure stationary uniformly optimal strategies which in turn implies that  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}}^{(k)})$  admits pure stationary optimal strategies. Since, by Shapley’s theorem,  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_k}^{(k)})$  has pure stationary optimal strategies, trivial backward induction on  $m$  will yield immediately Theorem 2.

**Proposition 2.** Let  $\mathcal{A}$  be a finite arena with states labelled by priorities from  $\{1, \dots, k\}$ . Let  $m \in \{1, \dots, k\}$  and  $\lambda_1, \dots, \lambda_{m-1}$  be a sequence of discount factors for priorities  $1, \dots, m$ , all belonging to the interval  $[0, 1]$ . Suppose that the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_m}^{(k)})$  has pure stationary strategies for both players. Then the following conditions hold:

- (i) for both players there exist pure stationary uniformly optimal strategies in the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)})$ ,
- (ii) there exists an  $\epsilon > 0$  such that, for each pair of pure stationary strategies  $(\sigma, \tau)$  for players 1 and 2, whenever  $\sigma$  and  $\tau$  are optimal in the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)})$  for some  $1 - \epsilon < \lambda_m < 1$  then  $\sigma$  and  $\tau$  optimal for all games  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)})$  with  $1 - \epsilon < \lambda_m < 1$ , in particular  $\sigma$  and  $\tau$  are uniformly optimal,

- (iii) if  $\sigma, \tau$  are pure stationary uniformly optimal strategies in the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_m}^{(k)})$  then they are optimal in the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}}^{(k)})$ ,
- (iv)  $\lim_{\lambda_m \uparrow 1} \text{val}_s(\mathcal{A}, u_{\lambda_1, \dots, \lambda_m}^{(k)}) = \text{val}_s(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}}^{(k)})$ , where  $\text{val}_s(\mathcal{A}, u_{\lambda_1, \dots, \lambda_m}^{(k)})$  is the value of the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_m}^{(k)})$  for an initial state  $s$ .

We precede the proof of Proposition 2 by auxiliary lemmas.

**Lemma 1.** *Suppose that  $\lambda_1, \dots, \lambda_k$ , the discount factors for all priorities, are strictly smaller than 1. Let  $\sigma, \tau$  be pure stationary strategies for players 1 and 2 in the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_k}^{(k)})$ . Then the expectation  $\mathbb{E}_{\sigma, \tau}^s[u_{\lambda_1, \dots, \lambda_k}^{(k)}]$  is a rational function of  $\lambda_1, \dots, \lambda_n$  bounded on  $[0, 1)^k$ .*

In fact, if we fix pure stationary strategies then we get a finite Markov chain with discounted evaluation. In this context Lemma 1 is standard, at least for one discount factor, see for example [14], and the extension to several discount factors is trivial.

The lack of space compels us to skip the the proof of the following (intuitively obvious) lemma:

**Lemma 2.** *Let  $f(x_1, \dots, x_k)$  be a rational function well-defined and bounded on  $[0, 1)^k$ . Then, for each  $0 \leq m < k$ , the iterated limit  $\lim_{x_{m+1} \uparrow 1} \dots \lim_{x_k \uparrow 1} f(x_1, \dots, x_k)$  exists and is finite. Moreover, for every fixed  $(x_1, \dots, x_{m-1}) \in [0, 1)^{m-1}$  there exists  $\epsilon > 0$  such that the one-variable mapping*

$$x_m \mapsto \lim_{x_{m+1} \uparrow 1} \dots \lim_{x_k \uparrow 1} f(x_1, \dots, x_{m-1}, x_m, x_{m+1}, \dots, x_k)$$

is rational on the interval  $[1 - \epsilon, 1)$ .

For any infinite history  $h \in \mathcal{H}^\omega$  the value  $u_{\lambda_1, \dots, \lambda_m}^{(k)}(h)$  can be seen as a function of discount factors  $\lambda_1, \dots, \lambda_m$ . It turns out that

**Lemma 3.** *For each  $m \in \{1, \dots, k\}$  and for each  $h \in \mathcal{H}^\omega$ ,*

$$\lim_{\lambda_m \uparrow 1} u_{\lambda_1, \dots, \lambda_m}^{(k)}(h) = u_{\lambda_1, \dots, \lambda_{m-1}}^{(k)}(h) . \tag{13}$$

*Proof.* (Sketch) Let  $u_{\lambda_1, \dots, \lambda_m}^{(k)} = u_{\lambda_1, \dots, \lambda_m}^{\text{disc}} + u_{\lambda_1, \dots, \lambda_m}^{\text{lim}}$  be the decomposition of  $u_{\lambda_1, \dots, \lambda_m}^{(k)}$  onto the discount and limit parts. Let  $\lambda$  and  $\lambda^*$  be discount factor mappings from  $\{1, \dots, k\}$  into  $[0, 1]$  such that for  $i \in \{1, \dots, k\}$ ,  $\lambda(i) = \lambda_i$  for  $i \leq m$  and  $\lambda(i) = 1$  for  $i > m$ , while  $\lambda^*(i) = \lambda(i)$  for  $i \neq m$  and  $\lambda^*(i) = 1$  for  $i = m$ . As usually,  $\lambda_i = \lambda \circ \varphi_i$  and  $\lambda_i^* = \lambda^* \circ \varphi_i$  are the corresponding stochastic processes.

We examine three cases:

**Case 1:**  $m < \liminf_i \varphi_i(h)$

In this case, all priorities appearing infinitely often in the sequence  $\varphi_i(h), i = 0, 1, \dots$  have the corresponding discount factors equal to 1. Thus  $T(h) = \min\{j \mid$

$\lambda_l(h) = 1$  for all  $l \geq j$  is finite. Then  $\lim_{\lambda_m \uparrow 1} u_{\lambda_1, \dots, \lambda_m}^{\text{disc}}(h) = u_{\lambda_1, \dots, \lambda_{m-1}}^{\text{disc}}(h)$  since  $u_{\lambda_1, \dots, \lambda_m}^{\text{disc}}(h)$  is just a polynomial of variables  $\lambda_1, \dots, \lambda_m$ . Similarly,  $\lim_{\lambda_m \uparrow 1} u_{\lambda_1, \dots, \lambda_m}^{\text{lim}}(h) = u_{\lambda_1, \dots, \lambda_{m-1}}^{\text{lim}}(h)$ , and we get (13).

**Case 2:**  $m = \liminf_i \varphi_i(h)$

Since for infinitely many  $i$ ,  $\lambda_i(h) = \lambda_m < 1$ , we have  $\prod_{i=0}^\infty \lambda_i(h) = 0$ , and then  $u_{\lambda_1, \dots, \lambda_m}^{\text{lim}}(h) = 0$ .

Let  $T_0(h) := \max_j \{\varphi_j(h) < m\}$  be the last moment when a priority strictly smaller than  $m$  appears in the sequence  $\varphi_i(h), i \in \mathbb{N}$ , of visited priorities. Notice that  $T_0(h) < \infty$  and  $\sum_{0 \leq l \leq T_0(h)} \lambda_0(h) \cdots \lambda_{l-1}(h)(1 - \lambda_l(h))r_l(h) \xrightarrow{\lambda_m \uparrow 1} u_{\lambda_1, \dots, \lambda_{m-1}}^{\text{disc}}(h)$ . On the other hand,  $\sum_{l=T_0(h)+1}^\infty \lambda_0(h) \cdots \lambda_{l-1}(h)(1 - \lambda_l(h))r_l(h) = (\prod_{j=0}^{T_0(h)} \lambda_j(h)) \cdot \sum_{l=0}^\infty (\lambda_m)^l (1 - \lambda_m)r(m) = (\prod_{j=0}^{T_0(h)} \lambda_j(h)) \cdot r(m) \xrightarrow{\lambda_m \uparrow 1} (\prod_{j=0}^{T_0(h)} \lambda_j^*(h))r(m) = (\prod_{j=0}^\infty \lambda_j^*(h))r(\liminf_i \varphi_i(h)) = u_{\lambda_1, \dots, \lambda_{m-1}}^{\text{lim}}(h)$ . Thus we have shown that  $\lim_{\lambda_m \uparrow 1} u_{\lambda_1, \dots, \lambda_m}^{\text{disc}}(h) = u_{\lambda_1, \dots, \lambda_{m-1}}^{\text{disc}}(h) + u_{\lambda_1, \dots, \lambda_{m-1}}^{\text{lim}}(h)$ .

**Case 3:**  $m > \liminf_i \varphi_i(h)$

Since  $m > m - 1 \geq \liminf_i \varphi_i(h)$  both  $u_{\lambda_1, \dots, \lambda_m}^{\text{lim}}(h)$  and  $u_{\lambda_1, \dots, \lambda_{m-1}}^{\text{lim}}(h)$  are equal to 0. Thus it suffices to show that

$$\lim_{\lambda_m \uparrow 1} u_{\lambda_1, \dots, \lambda_m}^{\text{disc}}(h) = u_{\lambda_1, \dots, \lambda_{m-1}}^{\text{disc}}(h) \tag{14}$$

For a subset  $Z$  of  $\mathbb{N}$  set  $f_Z(\lambda_1, \dots, \lambda_m) = \sum_{i \in Z} (1 - \lambda_i(h))\lambda_0(h) \cdots \lambda_{i-1}(h)r_i(h)$  and consider  $f_X(\lambda_1, \dots, \lambda_m)$  and  $f_Y(\lambda_1, \dots, \lambda_m)$ , where  $X = \{i \mid \varphi_i(h) = m\}$  and  $Y = \mathbb{N} \setminus X$ . We show that

$$\lim_{\lambda_m \uparrow 1} f_X(\lambda_1, \dots, \lambda_m) = 0 \tag{15}$$

This is obvious if  $X$  is finite, thus assume  $X$  infinite. Define a process  $T_i$ :  $T_0(h) = -1, T_{i+1}(h) = \min\{j \mid j > T_i(h) \text{ and } \varphi_j(h) = m\}$ . Thus  $T_i(h), i = 1, 2, \dots$ , gives the time of the  $i$ -th visit to a state with priority  $m$ . Set  $p(h) = \liminf_i \varphi_i(h)$  and define another process<sup>2</sup>:  $W_i(h) = \sum_{j=0}^{T_i(h)-1} \mathbf{1}_{\{\varphi_j(h)=p(h)\}}$ . Thus  $W_i(h)$  gives the number states with priority  $p(h)$  that were visited prior to the moment  $T_i(h)$ . Notice that, for all  $i \geq 1, \lambda_0(h) \dots \lambda_{T_i(h)-1}$  contains  $i - 1$  factors  $\lambda_m$  and  $W_i(h)$  factors  $\lambda_{p(h)}$  (and possibly other discount factors) whence  $\lambda_0(h) \dots \lambda_{T_i(h)-1} \leq (\lambda_m)^{i-1} (\lambda_{p(h)})^{W_i(h)}$  implying  $f_X(\lambda_1, \dots, \lambda_m) = (1 - \lambda_m)r(m) \sum_{i=0}^\infty \lambda_0(h) \dots \lambda_{T_i(h)-1}(h) \leq (1 - \lambda_m)r(m) \sum_{i=0}^\infty (\lambda_{p(h)})^{W_{i+1}(h)} (\lambda_m)^{i-1}$ . Now notice that  $\lim_{i \rightarrow \infty} W_i(h) = \infty$  since  $p(h)$  is visited infinitely often in  $h$ . Since  $p(h) < m$ , we have  $\lambda_{p(h)} < 1$  and  $\lim_{i \rightarrow \infty} (\lambda_{p(h)})^{W_{i+1}(h)} = 0$ . This implies easily (15) (applying the well-know and easy fact that summable series are Abel summable, [15]).

<sup>2</sup> We use the usual notation,  $\mathbf{1}_A$  is the indicator function of an event  $A, \mathbf{1}_A(h) = 1$  if  $\mathcal{H}^\omega \ni h \in A$  and  $\mathbf{1}_A(h) = 0$  otherwise.

Now let us examine  $f_Y(\lambda_1, \dots, \lambda_m)$ . Note that  $f_Y(\lambda_1, \dots, \lambda_{m-1}, 1) = u_{\lambda_1, \dots, \lambda_{m-1}}^{\text{disc}}(h)$ . Then  $\lim_{\lambda_m \uparrow 1} f_Y(\lambda_1, \dots, \lambda_m) = f_Y(\lambda_1, \dots, \lambda_{m-1}, 1)$  follows directly from the well-know Abel’s lemma for power series, see [15]. This and (15) yield (14).  $\square$

*Proof of Proposition 2.* Since the payoff mappings  $u_{\lambda_1, \dots, \lambda_{i+1}}^{(k)}$  are bounded and Borel-measurable, Lebesgue’s dominated convergence theorem and Lemma 3 imply that for all strategies  $\sigma$  and  $\tau$  for players 1 and 2,  $\lim_{\lambda_{i+1} \uparrow 1} \mathbb{E}_{\sigma, \tau}^s(u_{\lambda_1, \dots, \lambda_{i+1}}^{(k)}) = \mathbb{E}_{\sigma, \tau}^s(\lim_{\lambda_{i+1} \uparrow 1} u_{\lambda_1, \dots, \lambda_{i+1}}^{(k)}) = \mathbb{E}_{\sigma, \tau}^s(u_{\lambda_1, \dots, \lambda_i}^{(k)})$ . Iterating, we get

$$\lim_{\lambda_{m+1} \uparrow 1} \dots \lim_{\lambda_k \uparrow 1} \mathbb{E}_{\sigma, \tau}^s(u_{\lambda_1, \dots, \lambda_k}^{(k)}) = \mathbb{E}_{\sigma, \tau}^s(u_{\lambda_1, \dots, \lambda_m}^{(k)}) . \tag{16}$$

Suppose that strategies  $\sigma$  and  $\tau$  are pure stationary. Then, by Lemma 1, the mapping  $[0, 1]^k \ni (\lambda_1, \dots, \lambda_k) \mapsto \mathbb{E}_{\sigma, \tau}^s(u_{\lambda_1, \dots, \lambda_k}^{(k)})$  is rational and bounded. Lemma 2 applied to the left hand side of (16) allows us to deduce that, for fixed  $\lambda_1, \dots, \lambda_{m-1}$ , the mapping

$$(0, 1) \ni \lambda_m \mapsto \mathbb{E}_{\sigma, \tau}^s(u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)}) \tag{17}$$

is a rational mapping (of  $\lambda_m$ ) for  $\lambda_m$  sufficiently close to 1.

For pure stationary strategies  $\sigma$  and  $\sigma^\#$  for player 1 and  $\tau, \tau^\#$  for player 2 and fixed discount factors  $\lambda_1, \dots, \lambda_{m-1}$  we consider the mapping

$$[0, 1) \ni \lambda_m \mapsto \Phi_{\sigma^\#, \tau^\#, \sigma, \tau}(\lambda_m) := \mathbb{E}_{\sigma^\#, \tau^\#}^s(u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)}) - \mathbb{E}_{\sigma, \tau}^s(u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)}) .$$

As a difference of rational mappings, all mappings  $\Phi_{\sigma^\#, \tau^\#, \sigma, \tau}$  are rational for  $\lambda_m$  sufficiently close to 1. Since rational mappings are continuous and have finitely many zeros, for each  $\Phi_{\sigma^\#, \tau^\#, \sigma, \tau}$  we can find  $\epsilon > 0$  such that  $\Phi_{\sigma^\#, \tau^\#, \sigma, \tau}$  does not change the sign for  $1 - \epsilon < \lambda_m < 1$ , i.e.  $\forall \lambda_m \in (1 - \epsilon, 1)$ ,

$$\Phi_{\sigma^\#, \tau^\#, \sigma, \tau}(\lambda_m) \geq 0, \quad \text{or} \quad \Phi_{\sigma^\#, \tau^\#, \sigma, \tau}(\lambda_m) = 0, \quad \text{or} \quad \Phi_{\sigma^\#, \tau^\#, \sigma, \tau}(\lambda_m) \leq 0 . \tag{18}$$

Moreover, since there is only a finite number of pure stationary strategies, we can choose in (18) the same  $\epsilon$  for all mappings  $\Phi_{\sigma^\#, \tau^\#, \sigma, \tau}$ , where  $\sigma, \sigma^\#$  range over pure stationary strategies of player 1 while  $\tau, \tau^\#$  range over pure stationary strategies of player 2.

Suppose that  $\sigma^\#, \tau^\#$  are optimal pure stationary strategies in the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)})$  for some  $\lambda_m \in (1 - \epsilon, 1)$ . This means that for all strategies  $\sigma, \tau$  for both players

$$\mathbb{E}_{\sigma, \tau^\#}^s(u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)}) \leq \mathbb{E}_{\sigma^\#, \tau^\#}^s(u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)}) \leq \mathbb{E}_{\sigma^\#, \tau}^s(u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)}) . \tag{19}$$

For pure stationary strategies  $\sigma, \tau$ , Eq. (19) is equivalent with  $\Phi_{\sigma^\#, \tau^\#, \sigma, \tau^\#}(\lambda_m) \geq 0$  and  $\Phi_{\sigma^\#, \tau^\#, \sigma^\#, \tau}(\lambda_m) \leq 0$ . However, if these two inequalities are satisfied for some  $\lambda_m$  in  $(1 - \epsilon, 1)$  then they are satisfied for all such  $\lambda_m$ , i.e. (19) holds for all  $\lambda_m$  in  $(1 - \epsilon, 1)$  for all all pure stationary strategies  $\sigma, \tau$ . (Thus we have proved that

$\sigma^\#$  and  $\tau^\#$  are optimal for all  $\lambda_m \in (1 - \epsilon, 1)$  but only if we restrict ourselves to the class of pure stationary strategies.) But we have assumed that for each  $\lambda_m$  the game  $(\mathcal{A}, u_{\lambda_1, \dots, \lambda_{m-1}, \lambda_m}^{(k)})$  has optimal pure stationary strategies (now we take into account all strategies), and under this assumption it is straightforward to prove that if (19) holds for all pure stationary strategies  $\sigma, \tau$  then it holds for all strategies  $\sigma, \tau$ , i.e.  $\sigma^\#$  and  $\tau^\#$  are optimal in the class of all strategies and for all  $\lambda_m \in (1 - \epsilon, 1)$ . In this way we have proved conditions (i) and (ii) of Proposition 2.

Applying the limit  $\lambda_m \uparrow 1$  to (19) and taking into account (16) we get

$$\mathbb{E}_{\sigma, \tau^\#}^s(u_{\lambda_1, \dots, \lambda_{m-1-1}, \lambda_{m-1}}^{(k)}) \leq \mathbb{E}_{\sigma^\#, \tau^\#}^s(u_{\lambda_1, \dots, \lambda_{m-1-1}, \lambda_{m-1}}^{(k)}) \leq \mathbb{E}_{\sigma^\#, \tau}^s(u_{\lambda_1, \dots, \lambda_{m-1\epsilon-1}, \lambda_{m-1\epsilon}}^{(k)}),$$

which proves (iii). It is obvious that this implies also (iv).  $\square$

## References

1. Shapley, L.S.: Stochastic games. *Proceedings Nat. Acad. of Science USA* 39, 1095–1100 (1953)
2. Emerson, E., Jutla, C.: Tree automata,  $\mu$ -calculus and determinacy. In: FOCS 1991, pp. 368–377. IEEE Computer Society Press, Los Alamitos (1991)
3. de Alfaro, L., Henzinger, T.A., Majumdar, R.: Discounting the future in systems theory. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 1022–1037. Springer, Heidelberg (2003)
4. de Alfaro, L., Majumdar, R.: Quantitative solution to omega-regular games. *Journal of Computer and System Sciences* 68, 374–397 (2004)
5. Gimbert, H., Zielonka, W.: Deterministic priority mean-payoff games as limits of discounted games. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052(Part II), pp. 312–323. Springer, Heidelberg (2006)
6. Chatterjee, K., Jurdziński, M., Henzinger, T.: Quantitative stochastic parity games. In: Proceedings of the 15th Annual Symposium on Discrete Algorithms SODA, pp. 114–123 (2004)
7. McIver, A., Morgan, C.: Games, probability and the quantitative  $\mu$ -calculus qmu. In: Baaz, M., Voronkov, A. (eds.) LPAR 2002. LNCS (LNAI), vol. 2514, pp. 292–310. Springer, Heidelberg (2002), Full version [arxiv.org/abs/cs.L0/0309024](https://arxiv.org/abs/cs.L0/0309024)
8. de Alfaro, L.: Formal Verification of Probabilistic Systems. PhD thesis, Stanford University (December 1997)
9. Zielonka, W.: Perfect-information stochastic parity games. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 499–513. Springer, Heidelberg (2004)
10. Osborne, M.J., Rubinstein, A.: *A Course in Game Theory*. MIT Press, Cambridge (2002)
11. Filar, J., Vrieze, K.: *Competitive Markov Decision Processes*. Springer, Heidelberg (1997)
12. McIver, A., Morgan, C.: A novel stochastic game via the quantitative mu-calculus. In: Cerone, A., Wiklicky, H. (eds.) QAPL 2005. Proc. of the Third Workshop on Quantitative Aspects of Programming Languages. ENTCS, vol. 153(2), pp. 195–212. Elsevier, Amsterdam (2005)
13. Blackwell, D.: Discrete dynamic programming. *Annals of Mathematical Statistics* 33, 719–726 (1962)
14. Hordijk, A., Yushkevich, A.: Blackwell optimality. In: Feinberg, E., Schwartz, A. (eds.) *Handbook of Markov Decision Processes*, Kluwer, Dordrecht (2002)
15. Titchmarsh, E.: *The Theory of Functions*, 2nd edn. Oxford University Press, Oxford

# Bounded Depth Data Trees

Henrik Björklund<sup>1,\*</sup> and Mikołaj Bojańczyk<sup>2,\*\*</sup>

<sup>1</sup> University of Dortmund

<sup>2</sup> Warsaw University

**Abstract.** A data tree is a tree where each node has a label from a finite set, and a data value from a possibly infinite set. We consider data trees whose depth is bounded beforehand. By developing an appropriate automaton model, we show that under this assumption various formalisms, including a two variable first-order logic and a subset of XPath, have decidable emptiness problems.

## 1 Introduction

A data tree is a tree where each node has a label from a finite set, and a data value from a possibly infinite set. We consider trees where there is a fixed bound on the depth of nodes. For recognizing properties of such trees, we define an automaton model that traverses the trees in a depth-first manner. We show that the emptiness problem for the automata is decidable, by a reduction to reachability for priority multicounter automata, a powerful model for recognizing word languages [32]. The automaton model is used to show decidability of the satisfiability problem for a two-variable first-order logic, and also for a fragment of XPath. In the logic and XPath, we allow a rich vocabulary of navigational predicates, such as document order, thus extending the work from [6], where only successor axes were allowed.

The main application area for this paper is static analysis tasks for XML databases. We would like to develop tools that automatically answer questions such as: does property  $a$  of XML documents always imply property  $b$ ?; or: is property  $a$  vacuously true?

A very successful approach to static analysis has been to use tree automata. An XML document is modeled as a tree, where the labels of the tree correspond to tag names in the document. Many formalisms for XML can be represented as tree automata, possibly extended with additional features, (see, e.g., [29,23]). Using this representation, a large body of techniques for tree automata can be applied to solving static analysis tasks.

A drawback of the tree automaton approach is that it considers only the tag names, and ignores other content stored in the document. For instance, one cannot express key constraints such as: “every two nodes have different values

---

\* Supported by the Deutsche Forschungsgemeinschaft Grant SCHW678/3-1 and the DAAD Grant D/05/02223.

\*\* Supported by Polish government grant no. N206 008 32/0810.

stored in their `unique_key` attribute”. Such constraints are clearly important for databases, and can be expressed in, say, XPath. One way of extending the tree automata approach beyond mere tag names is to consider data trees. In a data tree, each node has a label from a finite set, and a data value from a possibly infinite set. The data values are used to model the content of the document. Recently, there has been flurry of research on models with data, including data words [19,31,13,14,5,22,3], and data trees [8,14,16,6].

The typical tool for solving logics over trees without data is a finite-state automaton. When data is added, the appropriate automaton almost always involves counting: ranging from automata with semilinear constraints on runs [6], through vector-addition systems [5], and on to faulty counter machines [14] and lossy channel machines [22]. Complexities are often high: non-primitive recursive, e.g. [14] and some results in [22], or as hard as reachability in vector-addition systems [5] (a decidable problem, but not known to be primitive recursive [26,21]).

Due to the above, logics for objects with data are usually quite weak. For data trees, the present cutting edge is a fragment of first-order logic, where only two variables are used, and only the child and next-sibling axes are allowed for testing spatial relationship [6]. This logic has decidable emptiness, but most extensions are undecidable: adding a third variable, adding a second data value, adding order on data values. One question left open in [6] was whether the logic remains decidable if we add their transitive closures (i.e. descendant and following sibling)? The outlook is not optimistic, since the extended problem subsumes reachability for tree vector-addition systems, a difficult open problem [12].

In this paper, we try to deal with the extended axes. We show that if a fixed bound on the depth is imposed, the logic from [6] remains decidable after the descendant and following sibling axes (and even document order) are added to the signature. (The following sibling axis is more interesting than the descendant axis in bounded depth trees.) In terms of XPath, we extend the fragment from [6], by allowing all navigational axes in Core XPath ([17]) in path expressions, and considerably stronger predicate expressions, where the data values of two relative paths can be compared, as long as the paths belong to the same subtree.

Another motivation to consider bounded depth trees is that the lower bounds in [6] are somewhat artificial, using constructions alien to actual XML documents. Indeed, many XML data bases are wide, but not very deep (see, e.g., [9]). Therefore, considering trees of arbitrary depth, which turns out to be a major technical difficulty, need not reflect problems in the real world. It is, however, sometimes crucial to compare elements on a horizontal axis (which nodes are later/earlier in the document), something that cannot be done by the logic in [6].

What do we gain by bounding the depth? The main idea is that a bounded depth tree actually bears more resemblance to a word than a tree. If a bounded depth tree is written down as a string (the way XML documents are stored in text files), a finite string automaton can recover the tree structure by using its finite control to keep track of the path leading to the current node. This simple observation is the essence of our approach. However, it is not immediately clear how the string automaton should deal with data values. We discover that the



appropriate model is an extension of multicounter automata, where a limited form of zero test is allowed [32].

The paper is structured as follows. In Section 2 we define bounded depth data trees, and some notions for discussing them. Section 3 contains the main contributions of the paper. Here, we present our automaton model, describe its basic properties, and prove that the corresponding emptiness problem is decidable. Section 4 describes a fragment of first-order logic, which, thanks to the automaton model, can be shown to have a decidable satisfiability problem. Section 5 describes applications for XPath. Due to space limitations, many proofs have been omitted, and will appear in the full version of the paper.

## 2 Definitions

To simplify technicalities, we do not actually consider data trees, but data forests. Informally, a data forest is an ordered sequence of data trees. Formally, a *data forest* is a partial function

$$t : \mathbb{N}^+ \rightarrow \Sigma \times \Delta$$

with nonempty finite domain. The set  $\Sigma$  is called the *alphabet* and is required to be finite, while the set  $\Delta$  is called the *data domain*, and may be infinite (usually, we use the natural numbers for  $\Delta$ ). The *nodes* of the forest are elements of the domain of  $t$ . The first coordinate of  $t(x)$  is called *the label* of the node  $x$ , while the second coordinate of  $t(x)$  is called the *data value* of  $x$ . Furthermore, the set of nodes must be closed under parents and previous siblings:

- The *parent* of a node  $a_1 \cdots a_n a_{n+1} \in \mathbb{N}^*$  is the node  $a_1 \cdots a_n$ .
- The *previous sibling* of a node  $a_1 \cdots a_n \in \mathbb{N}^*$  is the node  $a_1 \cdots a_{n-1}(a_n - 1)$ . (A node with  $a_n = 0$  has no previous sibling.)

*Preceding siblings* are defined by taking the transitive closure the previous sibling. The opposite of previous/preceding siblings are next/following siblings. A node has at most one previous/next sibling, but possibly many preceding/following siblings. A *root* in a forest is any node without a parent; there may be many roots. The *depth* of a node  $a_1 \cdots a_n$  is the number  $n$ ; in particular each root has depth 1. The opposite of parent is *child*. The transitive closure of the child relation is the *descendant* relation, similarly *ancestors* are defined for parents. A *leaf* is a node without children.

A *depth  $k$  data forest* is one where all leaves have depth  $k$ . We could also consider forests where leaves have depth at most  $k$ ; however the more general type can be easily encoded in the special one by adding dummy nodes. When considering depth  $k$  data forests, we assume without loss of generality that the label set  $\Sigma$  is partitioned into  $k$  disjoint sets  $\Sigma_1, \dots, \Sigma_k$  such that nodes of depth  $i$  are only allowed to use labels from  $\Sigma_i$ . This assumption can be easily ensured by expanding the alphabet.

A *class* of a data forest is a maximal set of nodes with the same data value.

Let  $t$  be a forest. The *depth-first-search traversal* (DFS traversal) of  $t$  is a sequence  $v_1, \dots, v_n$  of nodes of  $t$  satisfying:

- Each non-leaf node appears twice, and each leaf appears once.
- If  $i < n$  and  $v_i$  appears for the first time, i.e.  $v_i \notin \{v_1, \dots, v_{i-1}\}$ , then  $v_{i+1}$  is the leftmost child of  $v_i$ , except if  $v_i$  is a leaf, in which case  $v_{i+1}$  is the next sibling of  $v_i$ , or, if  $v_i$  is a rightmost child, the parent of  $v_i$ .
- If  $i < n$  and  $v_i$  is seen for the second time, i.e.  $v_i \in \{v_1, \dots, v_{i-1}\}$ , then  $v_{i+1}$  is the next sibling of  $v_i$ , or the parent of  $v_i$  if  $v_i$  is a rightmost child.

There is only one DFS traversal, and it must begin with the leftmost root and end with the rightmost root. Later on, it will be convenient that non-leaf nodes are visited twice. If we remove repetitions from the DFS traversal (by deleting second occurrences), we get the *document ordering* on nodes of a forest.

### 3 Automata

This section contains the main contribution of the paper. In Section 3.1, we define an automaton model for bounded depth data forests. After showing some properties that can be recognized by our automata in Section 3.2, we show in Section 3.3 that the automata have decidable emptiness. The decidability proof is by reduction to reachability in an extended model of multicounter automata (Petri nets). Therefore, we have no primitive recursive upper bound for the complexity; lower bounds are also open.

The automaton model we define can be seen as an extension of the class memory automata for words from [3] to bounded depth forests. These are, in turn, a variant of the data automata from [5]. The basic idea is to use one class memory automaton per depth level in the forest.

#### 3.1 Class Memory Automata for Forests of Bounded Depth

A depth  $k$  forest class memory automaton ( $k$ -FCMA) is defined as follows. It has  $k + 1$  state spaces:  $Q, Q_1, \dots, Q_k$ . Each has an initial and a final subset:

$$I, F \subseteq Q \quad I_1, F_1 \subseteq Q_1, \quad \dots \quad I_k, F_k \subseteq Q_k .$$

The idea is that the states  $Q_i$  will be used to examine data values of nodes at depth at least  $i$ , while the states in  $Q$  are used to examine properties that do not involve data.

The automaton runs on an input depth  $k$  forest by visiting its nodes in the DFS sequence (in particular, non-leaf nodes are visited twice). At every moment of its run, it keeps its current state  $q \in Q$  – called the *global state* – as well as  $k$  *class memory functions* of the form

$$f_1 : \Delta \rightarrow Q_1 \quad \dots \quad f_k : \Delta \rightarrow Q_k .$$

Therefore, a configuration of the automaton consists of: a node  $v$  of the forest  $t$ , the global state  $q$  and the class memory functions  $f_1, \dots, f_k$ . (Thanks to the class memory functions, the automaton is a type of infinite-state system,

which contributes to the high complexity of emptiness. Each configuration can be finitely represented, since the class memory functions have finite non-initial support.) At the beginning of the run,  $v$  is the leftmost root,  $q$  is set to be a designated *initial state*  $q_I \in Q$ , while all the class memory functions  $f_1, \dots, f_k$  assign initial states to all data values  $d \in \Delta$ . (If there are many initial states, this produces nondeterminism.)

A single step of the automaton works as follows. Assume that the automaton is in a node  $v$  of depth  $i$  with data value  $d$ . Depending on the global state, the values of  $f_1(d), \dots, f_i(d)$ , and the label of  $v$ , the automaton picks a new global state and new values of  $f_1(d), \dots, f_i(d)$ . It then advances to the next node in the DFS traversal. Therefore, the transition function is a set of rules from

$$\bigcup_{i=1, \dots, k} Q \times Q_1 \times \dots \times Q_i \times \Sigma_i \times Q \times Q_1 \times \dots \times Q_i$$

Note that since  $\Sigma$  is partitioned into sets  $\Sigma_1, \dots, \Sigma_k$ , the label of a node determines its depth. In particular, the automaton knows if it is descending into a successor, moving to the right sibling, or ascending into the parent.

Furthermore, when the automaton has just read for the second time a rightmost sibling  $v$  at depth  $i \in \{1, \dots, k\}$  (or for the first time, if  $v$  is a leaf), it does some further processing on the class memory function  $f_i$ , which we call a *check-reset*. (The check-reset is done after the transition corresponding to the second visit in  $v$  has been applied.) First, the automaton checks if the class memory function  $f_i$  is *accepting*, i.e. all data values are assigned either initial or accepting states. If this is not the case, the run is aborted and cannot be continued. If this check succeeds, the class memory function  $f_i$  is reset, by assigning the initial state (nondeterministically, if there is more than one) to all data values.

The automaton accepts the data forest if, after completing the DFS traversal, it has an accepting global state (and the last-check reset has been successful). Note however, that before this happens, a large number of memory check-resets must be successfully carried out.

*Example 1.* Consider the following property of depth  $k$  forests: each data value occurs at most once. To recognize this property, the automaton only uses the states  $Q_1$  (all other state spaces  $Q$  and  $Q_2, \dots, Q_k$  contain one state  $q$ , which is both initial and final, and is never modified). There are two states in  $Q_1$ : an initial state *new* and a final state *old*. The transition function advances *new* to *old*, while *old* has no outgoing transitions. In other words, there is only one transition for each letter  $a \in \Sigma$ :

$$(q, \textit{new}, q, \dots, q, a, q, \textit{old}, q \dots, q).$$

### 3.2 Some Properties of FCMA

In this section we present some properties of bounded depth data forests that can be recognized by FCMA. Apart from being useful later on, the results in this section are meant to give a feeling for what FCMA can do.

**Fact.** FCMA are closed under union and intersection.

When the depth of a data forest is limited to 1, the forest is a *data word*, as considered in [5]. Furthermore, *data automata*, the automaton model introduced in [5] to recognize properties of data words, coincides with the restriction of FCMA to depth 1. Lemma 1 below can be used to transfer results about data words to data forests.

**Lemma 1.** *Let  $\mathcal{A}$  be a data automaton. The following properties of data forests are recognized by FCMA:*

- *For every node  $v$ , the children of  $v$ , when listed from left to right, form a data word accepted by  $\mathcal{A}$ .*
- *For every node  $v$ , the descendants of  $v$ , when listed in document order, form a data word accepted by  $\mathcal{A}$ .*

Sometimes it is convenient to see how the data value of a node is related to the data values of its neighborhood. The *profile* of a node is information about which nodes among its ancestors, previous and next siblings have the same data value. Once the depth  $k$  of forests is fixed, there are finitely many possible profiles. The following lemma shows that these can be tested by an automaton:

**Lemma 2.** *For each possible profile  $p$ , there is an FCMA that recognizes the language: “a node has label  $a$  if and only if it has profile  $p$ ”.*

We will also need to use FCMA to recognize languages of the form: “for every class, a given property holds”. Here we present a general result of this type. Note that it is not clear what we mean when saying that a class satisfies some property, since it is not clear how the nodes of a class should be organized once they are taken out of the data forest. Here we use one such definition, which we call a *take-out*. Let  $t$  be a forest and  $V$  a set of nodes in this forest. The nodes of the take-out are nodes of  $V$ , along with their ancestors. The labels in the take-out are inherited from  $t$ , except we add a special marker to distinguish if a node is from  $V$ , or just an ancestor of a node from  $V$ . The take-out is a forest without data, where leaves may have different depths.

**Lemma 3.** *Let  $L$  be a regular forest language (without data). An FCMA can test if the take-out of every class belongs to  $L$ .*

### 3.3 Decidable Emptiness for the Automata

In this section, we will show that emptiness is decidable for  $k$ -FCMA. The proof is by reduction to emptiness of priority multicounter automata. Note that universality is undecidable even for 1-FCMA, as it is already undecidable for data automata over words.

**Priority Multicounter Automata.** A priority multicounter automaton is an automaton over words (without data) that has a number of counters, which can be incremented, decremented and tested for zero. (Multicounter automata

with zero tests correspond to Petri nets with inhibitor arcs.) To keep the model decidable, the zero tests are restricted. This is where the priorities come in.

More formally, a *priority multicounter automaton* has a set  $C$  of counters, a state space  $Q$  and an input alphabet  $\Sigma$ . Furthermore, the counters come with a distinguished chain of subsets:  $C_1 \subseteq \dots \subseteq C_m \subseteq C$ .

The automaton reads a word  $w \in \Sigma^*$  from left to right, possibly using  $\epsilon$ -transitions. At each point in its run, the automaton has a current state  $q \in Q$  and a non-negative counter assignment  $c \in \mathbb{N}^C$ . At the beginning, a designated initial state is used, and all the counters are empty.

In a transition, the automaton reads a letter – possibly  $\epsilon$  – from the word. Depending on this letter the automaton changes its state, and performs a *counter operations*, that is, it increases a counter, decrements a counter, or checks that all counters in  $C_i$ , for some  $i$ , are empty.

The above operations can fail: if a decrement is done on an empty counter; or if a zero test fails. When the counter operation fails, the transition fails and the run is aborted. The automaton accepts if at the end of the word it has reached a designated accepting state. The following difficult result has been shown in [32]:

**Theorem 1.** *Emptiness is decidable for priority multicounter automata.*

Note that priority multicounter automata are an extension of multicounter automata (where the zero tests are not allowed). In particular, no primitive recursive emptiness algorithm is known.

**Reduction to Priority Multicounter Automata.** We now show that emptiness for FCMA can be reduced to emptiness of priority multicounter automata. In particular, thanks to Theorem 1, emptiness is decidable for FCMA.

Let  $t$  be a depth  $k$  forest, and let  $v_1, \dots, v_n$  be its DFS traversal. Let  $\text{trav}(t)$  be the word over  $\Sigma$  containing the labels of  $v_1, \dots, v_n$ . Since  $\text{trav}(t)$  does not use the data values, it is irrelevant if  $t$  is a data forest or a non-data forest.

**Theorem 2.** *Emptiness is decidable for  $k$ -FCMA, for all  $k \in \mathbb{N}$ . Furthermore, for each  $k$ -FCMA  $\mathcal{A}$ , the set  $\{\text{trav}(t) : t \text{ is accepted by } \mathcal{A}\}$  is accepted by an (effectively obtained) priority multicounter automaton.*

By Theorem 1, the first clause of the theorem follows from the second one. This section is therefore devoted to simulating a  $k$ -FCMA with a priority multicounter automaton.

We fix a  $k$ -FCMA  $\mathcal{A}$ . We assume that in every transition

$$(q, q_1, \dots, q_i, a, r, r_1, \dots, r_i),$$

none of the states  $r_1, \dots, r_i$  are initial; and if some  $q_j$  is initial, then so are  $q_{j+1}, \dots, q_i$ . Any  $k$ -FCMA can be effectively transformed into one satisfying the above assumptions.

The priority multicounter automaton that recognizes the traversals is defined as follows. It is a conjunction of two automata. The first one checks that the

depths indicated by the labels are consistent with a DFS traversal, i.e. the input word belongs to  $\{\text{trav}(t) : t \text{ is a depth } k \text{ forest}\}$ . Since the latter is a regular word language, we do not even need to use counters.

The real work is done by the second automaton, which we call  $\mathcal{B}$ . To simplify presentation, we use a slightly extended notion of transition. We will later comment on how the extended notion can be realized by a standard priority multicounter automaton. The control states of  $\mathcal{B}$  are the global states  $Q$  of  $\mathcal{A}$ . It has a counter for each of the states in  $Q_1, \dots, Q_k$  used in the class memory functions (we assume these state spaces are disjoint).

When the simulating automaton  $\mathcal{B}$  is in state  $q \in Q$ , and the input letter is  $a \in \Sigma_i$  (with  $i = 0, \dots, k$ ) the automaton performs the following actions:

1. As preprocessing for the transition,  $\mathcal{B}$  may nondeterministically choose to increment any counter corresponding to an initial state.
2. In the next step,  $\mathcal{B}$  nondeterministically picks a transition

$$(q, q_1, \dots, q_i, a, r, r_1, \dots, r_i)$$

of the simulated  $k$ -FCMA  $\mathcal{A}$ . It decrements counters  $q_1, \dots, q_i$ , and then increments the counters  $r_1, \dots, r_i$ .

3. In the third step,  $\mathcal{B}$  sets its finite control to the the state  $r$  from the transition chosen in step 2.
4. The last step corresponds to the check-reset and is carried out if the next label is going to be from  $\Sigma_{i-1}$  (this corresponds to a rightmost successor node appearing for the second time in the DFS, or for the first time, if the node is a leaf). In this case, the automaton  $\mathcal{B}$  tests that all counters in

$$Q_i \setminus (F_i \cup I_i) \tag{1}$$

are empty, and then empties all the counters in  $Q_i$ .

We call such a sequence of actions a *macrotransition*. A macrotransition can be carried out by a multicounter automaton with zero checks, by using  $\epsilon$ -transitions and additional control states. Perhaps the most delicate point is the last step in the macrotransition. First of all, the automaton needs to know the next label. Here, we can nondeterministically guess the next label in advance; this nondeterministic guess is then validated in the next step. (The degenerate case of  $j = 0$  is handled by using  $\epsilon$ -transitions.)

At first glance, the automaton is not a priority multicounter automaton, since the zero checks in (1) are done for disjoint counters. But this can easily be fixed, by imposing a chain discipline on the zero checks. Indeed, when the automaton is doing the zero check in (1), we know that in the previous moves it has emptied the counters  $Q_{i+1}, \dots, Q_k$ . Therefore, it could equivalently zero check the counters

$$Q_i \setminus (F_i \cup I_i) \cup Q_{i+1} \cup \dots \cup Q_k .$$

Furthermore, the emptying of the counters in  $Q_i$ , which is done after (1), can be simulated by a sequence of nondeterministic decrements on  $Q_i$  and then a

zero check on  $Q_i \cup \dots \cup Q_k$ . The automaton  $\mathcal{B}$  accepts if it reaches an accepting global state after processing all the nodes. It is fairly clear that if  $\mathcal{A}$  accepts  $t$ , then  $\mathcal{B}$  accepts  $\text{trav}(t)$ . Theorem 2 then follows once we show the converse:

**Lemma 4.** *If  $t$  is a depth  $k$  forest whose DFS traversal is accepted by  $\mathcal{B}$ , then  $t$  can be labeled with data values so that the resulting data forest is accepted by  $\mathcal{A}$ .*

**Proof**

Consider an accepting run of  $\mathcal{B}$ , with macrotransitions  $m_1, \dots, m_n$ . Let  $v_1, \dots, v_n$  be the DFS traversal of the forest  $t$ . These nodes correspond to the macrotransitions  $m_1, \dots, m_n$ . Recall that each macrotransition corresponds (in step 2) to a transition of the automaton  $\mathcal{A}$ . Let then  $\delta_1, \dots, \delta_n$  be the sequence of transitions of  $\mathcal{A}$  that corresponds to  $m_1, \dots, m_n$ .

We will assign data values to nodes of the forest  $t$ , so that the result  $s$  is accepted by  $\mathcal{A}$ , using the run  $\delta_1, \dots, \delta_n$ . This is done progressively for  $v_1, \dots, v_n$ , so that at each intermediate step  $j = 0, \dots, n$  the following invariant is satisfied.

Assume that data values have been assigned to nodes  $v_1, \dots, v_j$ . The sequence  $\delta_1, \dots, \delta_j$  is a partial run of  $\mathcal{A}$  on  $t$  (that has read nodes  $v_1, \dots, v_j$ ) such that:

For each class memory function  $f_i \in \{f_1, \dots, f_k\}$ , and each non-initial state  $q \in Q_i$ , the counter  $q$  contains the number of data values  $d$  with  $f_i(d) = q$ .

This invariant can be easily shown by induction on  $j$ . □

## 4 A Two-Variable Logic for Bounded Depth Data Forests

In this section, we define a first-order logic that can express properties of data forests. Formulas of this logic can be effectively compiled into FCMA; in particular this logic has decidable satisfiability thanks to Theorem 2. Variables quantify over nodes. Only two variables,  $x$  and  $y$ , are allowed. Furthermore, data values can only be compared for equality, via a predicate  $x \sim y$ . On the other hand, we allow a large body of navigational predicates.

For a fixed depth  $k$ , we define the logic  $FO_k^2$  as the two-variable fragment of FO, with the following predicates (some parameterized by  $i = 1, \dots, k$ ):

- $d_i(x)$   $x$  has depth  $i$
- $a(x)$   $x$  has label  $a$  (here  $a$  is a label from  $\Sigma$ )
- $x \downarrow_i y$   $y$  is a descendant of  $x$  and  $\text{depth}(y) - \text{depth}(x) = i$
- $x \downarrow_+ y$   $y$  is a descendant of  $x$
- $x + 1 = y$   $x$  is the left sibling of  $y$
- $x < y$   $x$  comes before  $y$  in the document ordering (the ordering produced by a pre-order traversal)
- $x < y$   $x$  and  $y$  are siblings, and  $x$  is to the left of  $y$
- $t_i(x, y)$   $x \preceq y$  and the nodes  $x, y$  share the same depth  $i$  ancestor but not the same depth  $i + 1$  ancestor

- $t_0(x, y)$   $x, y$  do not have a common ancestor
- $x \sim y$   $x$  and  $y$  have the same data value
- $x \oplus y$   $y$  is the *class successor* of  $x$ , that is,  $x \sim y$ ,  $x$  comes before  $y$  in the document ordering, and there is no  $z$  between  $x$  and  $y$  (in the document ordering) which has the same data value.

The semantic of the logic is defined as usual. For instance, the following is a long way of saying that all nodes have the same data value:

$$\forall x \forall y (x \downarrow_+ y \Rightarrow x \sim y) \wedge (x + 1 = y \Rightarrow x \sim y) .$$

The predicates  $d_i, \downarrow_+, \downarrow_i$  and  $<$  are syntactic sugar, and can be removed from the signature without loss of expressivity. For instance,  $d_i(x)$  is the same as  $t_i(x, x)$ . In similar ways,  $\downarrow_i, \downarrow_+$  can be defined in terms of  $t_i$ , and  $<$  can be defined in terms of  $t_i$  and  $<$ . Since we only have two variables,  $x + 1 = y$  cannot be defined in terms of  $<$ , and  $x \oplus y$  cannot be defined in terms of  $<$  and  $\sim$ .

In the following example, we show that thanks to the bounded depth assumption, two-variable formulas can express properties that seemingly require three variables.

*Example 2.* We write a formula  $\varphi(x)$ , which holds in a node  $x$  that has two distinct descendants  $y \sim z$ . The natural formula would be

$$\varphi(x) = \exists y \exists z (y \neq z \wedge y \sim z \wedge x \downarrow_+ y \wedge x \downarrow_+ z) .$$

The problem is that this formula uses three variables. We will show that for depth  $k$  forests,  $\varphi$  can be written with only two variables. The idea is to do a disjunction over the finitely many possible depths  $i$  of the node  $x$ :

$$\varphi(x) = \bigvee_i d_i(x) \wedge \exists y (x \downarrow_+ y \wedge \exists x (x \neq y \wedge x \sim y \wedge \bigvee_{j \geq i} t_j(x, y))) .$$

In the above formula, the second existential quantifier  $\exists x$  actually corresponds to the node  $z$ . We do not need to verify if the new node  $x$  is a descendant of the “real” node  $x$  in the free variable; this is a consequence of  $t_i(x, y)$ .

**Theorem 3.** *Every language definable in  $FO_k^2$  can be recognized by a  $k$ -FCMA.*

**Corollary 1.** *Satisfiability is decidable for  $FO_k^2$ .*

## 5 XPath

We now apply our results to show decidability of some static analysis tasks for XML. Our approach closely mirrors that in [6]. To avoid repetition, we only explain which expressive power can be *added* to the fragment *LocalDataXPath* from [6], while preserving decidability over bounded depth trees. Since we have the predicates  $\downarrow_+$  and  $<$  in our logic, we can, unlike [6], capture the XPath axes **descendant** (**ancestor**) and **following** (**preceding**). Also, we can allow



attribute comparisons in which *both* sides of the (in-)equality are *relative*, as long as they stay *within the subtree* rooted at the node to which they are relative.

As in [6], decidability is shown by encoding XPath into two-variable logic. We first give an example that illustrates how the bounded depth can be used to encode XPath expressions that could not be handled in [6]. This example is similar to Example 2 in the Section 4.

*Example 3.* Consider the XPath expression

$$\text{child} :: a/\text{child} :: b/@B_1 = \text{child} :: c/\text{next} - \text{sibling} :: d/@B_2.$$

It is not allowed in LocalDataXPath, since both sides of the equality are relative paths. For bounded depth trees, however, we can encode it into the logic from Section 4, using a technique similar to that of Example 2.

We now define *BDXPath* (Bounded Depth XPath). It is defined the same way as LocalDataXPath from [6]; except that BDXPath can use all the navigational axes in Core XPath [17]. Also, in predicate expressions, we allow comparisons of attribute values with  $=$  and  $\neq$  as long as one of the following holds.

1. At least one side of the (in-)equality is an *absolute* location path (i.e., one starting at the root, or document node); or
2. The comparison is relative, but *safe* (as defined in [6]); or
3. Both sides have location expressions that start with **child** or **descendant** and do not use **parent** or **ancestor**.

Using the same proof, but aided by our more powerful two-variable logic, we can upgrade the main XPath result from [6]:

**Theorem 4.** *Over trees of bounded depth, Satisfiability and Containment for (unary or binary) BDXPath is decidable. This holds even relative to a schema consisting of a regular tree language and unary key and inclusion constraints.*

*Acknowledgements.* We thank Wim Martens and Thomas Schwentick for valuable discussions.

## References

1. Alon, N., Milo, T., Neven, F., Suciu, D., Vianu, V.: XML with Data Values: Typechecking Revisited. In JCSS 66(4), 688–727 (2003)
2. Arenas, M., Fan, W., Libkin, L.: Consistency of XML specifications. In: Bertossi, L., Hunter, A., Schaub, T. (eds.) Inconsistency Tolerance. LNCS, vol. 3300, pp. 15–41. Springer, Heidelberg (2005)
3. Björklund, H., Schwentick, T.: On notions of regularity for data languages Manuscript (2006) available at <http://lrb.cs.uni-dortmund.de/~bjork/papers/regular-data.pdf>
4. Benedikt, M., Fan, W., Geerts, F.: XPath Satisfiability in the Presence of DTDs. In: PODS 2005

5. Bojańczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on words with data. In: LICS 2006, pp. 7–16 (2006)
6. Bojańczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-Variable Logic on Data Trees and XML Reasoning. In: PODS'06 (2006)
7. Bouyer, P., Petit, A., Thérien, D.: An algebraic approach to data languages and timed languages. *Inf. Comput.* 182(2), 137–162 (2003)
8. Buneman, P., Davidson, S.B., Fan, W., Hara, C.S., Tan, W.C.: Reasoning about keys for XML. In *Inf. Syst.* 28(8), 1037–1063 (2003)
9. Choi, B.: What are real DTDs like. In: WebDB 2002, pp. 43–48 (2002)
10. Cristau, J., Löding, C., Thomas, W.: Deterministic Automata on Unranked Trees. In: Liśkiewicz, M., Reischuk, R. (eds.) FCT 2005. LNCS, vol. 3623, pp. 68–79. Springer, Heidelberg (2005)
11. David, C.: Mots et données infinis. Master thesis, Université Paris 7, LIAFA (2004)
12. de Groot, P., Guillaume, B., Salvati, S.: Vector Addition Tree Automata. In: LICS 2004, pp. 64–73 (2004)
13. Demri, S., Lazic, R., Nowak, D.: On the Freeze Quantifier in Constraint LTL: Decidability and Complexity. In: TIME 2005
14. Demri, S., Lazic, R.: LTL with the Freeze Quantifier and Register Automata. In: LICS 2006, pp. 17–26 (2006)
15. Etessami, K., Vardi, M.Y., Wilke, T.: First-Order Logic with Two Variables and Unary Temporal Logic. *Inf. Comput.* 179(2), 279–295 (2002)
16. Geerts, F., Fan, W.: Satisfiability of XPath Queries with Sibling Axes. In: Bierman, G., Koch, C. (eds.) DBPL 2005. LNCS, vol. 3774, Springer, Heidelberg (2005)
17. Gottlob, G., Koch, C., Pichler, R.: Efficient Algorithms for Processing XPath Queries. In: Bressan, S., Chaudhri, A.B., Lee, M.L., Yu, J.X., Lacroix, Z. (eds.) CAISE 2002 and VLDB 2002. LNCS, vol. 2590, Springer, Heidelberg (2003)
18. Grädel, E., Otto, M.: On Logics with Two Variables. *TCS* 224, 73–113 (1999)
19. Kaminski, M., Francez, N.: Finite memory automata. *TCS* 134, 329–363 (1994)
20. Kieroński, E., Otto, M.: Small Substructures and Decidability Issues for First-Order Logic with Two Variables. In: LICS 2005 (2005)
21. Kosaraju, S.R.: Decidability of reachability in vector addition systems. In: STOC 1982, pp. 267–281 (1982)
22. Lazić, R.: Safely Freezing LTL. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, Springer, Heidelberg (2006)
23. Martens, W.: Static analysis of XML transformation and schema. PhD Thesis, Hasselt University (2006)
24. Martens, W., Niehren, J.: Minimizing Tree Automata for Unranked Trees. In: Bierman, G., Koch, C. (eds.) DBPL 2005. LNCS, vol. 3774, Springer, Heidelberg (2005)
25. Marx, M.: First order paths in ordered trees. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, Springer, Heidelberg (2004)
26. Mayr, E.: An algorithm for the general Petri net reachability problem. In: STOC 1981, pp. 238–246 (1981)
27. Mortimer, M.: On languages with two variables. *Zeitschr. f. math. Logik u. Grundlagen d. Math.* 21, 135–140 (1975)
28. Neeraj Verma, K., Seidl, H., Schwentick, T.: On the Complexity of Equational Horn Clauses. In: Nieuwenhuis, R. (ed.) Automated Deduction – CADE-20. LNCS (LNAI), vol. 3632, Springer, Heidelberg (2005)

29. Neven, F.: Automata, Logic, and XML. In: Bradfield, J.C. (ed.) CSL 2002 and EACSL 2002. LNCS, vol. 2471, pp. 2–26. Springer, Heidelberg (2002)
30. Neven, F., Schwentick, T.: XPath Containment in the Presence of Disjunction, DTDs, and Variables. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, Springer, Heidelberg (2002)
31. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.* 15(3), 403–435 (2004)
32. Reinhardt, K.: Counting as Method, Model and Task in Theoretical Computer Science. Habilitation-thesis (2005)
33. XML Path Language (XPath), W3C Recommendation (November 16, 1999) Available at <http://www.w3.org/TR/xpath>

# Unranked Tree Automata with Sibling Equalities and Disequalities

Wong Krianto and Christof Löding

Lehrstuhl für Informatik 7, RWTH Aachen, Germany

**Abstract.** We propose an extension of the tree automata with constraints between direct subtrees (Bogaert and Tison, 1992) to unranked trees. Our approach uses MSO-formulas to capture the possibility of comparing unboundedly many direct subtrees. Our main result is that the nonemptiness problem for the deterministic automata, as in the ranked setting, is decidable. Furthermore, we show that the nondeterministic automata are more expressive than the deterministic ones.

## 1 Introduction

The notion of unranked trees, i.e., finite (ordered) trees for which there are no constraints on the number of successors of a node, has recently regained interest from the research community, especially due to the application of such trees as models of semi-structured data. As with ranked trees, automata-related and logic-related notions have been developed for unranked trees. In fact, many results that hold for the ranked case have been shown to hold for the unranked case as well. For references, the reader is referred to, e.g., the surveys [17,18].

A current trend in the theory of unranked tree automata is concerned with the development of logics and automaton models that are more expressive than the framework of finite automata and, at the same time, have good (algorithmic) properties. Such frameworks, in turn, can be useful for developing (logic-based) query languages over unranked trees (with application to query languages for XML documents) with desirable algorithmic properties. A particular approach along this line, for instance, has been to incorporate the notion of numerical constraints in unranked tree automata: in the Presburger automata of Seidl et. al. [17,18] (cf. also the sheaves automata of Lugiez and Dal Zilio [8]), the applicability of a (bottom-up) transition at a node of an input tree is subject to formulas of Presburger arithmetic (the first-order logic over the natural numbers) over the occurrences of the states to which the children of this node are evaluated to. In another approach, structures (among others, words and trees) with data, i.e. where the nodes carry, besides a label from a finite alphabet, a data value from an infinite set, have been considered; see, e.g., [3,2] as well as the references therein. In these two papers, logics and automata with equality tests between data values in different nodes are studied, and decidability results for fragments of first-order logics over words and unranked trees with data are shown.

Regarding the latter approach, another possibility of incorporating data is to encode the data value of a node (together with the node's label) as a subtree over

a finite alphabet (e.g., natural numbers can be coded by unary subtrees of the corresponding depth) instead of directly taking values from an infinite alphabet. In this view, equality tests between data values then amount to equality comparisons between subtrees. As a first step toward this approach, in this paper we study a class of (unranked) tree automata that can deal with such comparisons in a restricted way.

In the ranked setting, automata with equality comparisons between subtrees have been studied in the literature; for references, see [7, Chapter 4]. It turns out that tree automata with such constraints, in the most general form where it is allowed to compare arbitrary subtrees, fail to have a decidable nonemptiness problem [15]. This result even carries over to the case where equality tests are only allowed between cousin subtrees, i.e., subtrees of depth at most two [19]. Nevertheless, by imposing appropriate restrictions on the transition structure and/or the equality constraints, it is possible to identify classes of automata with a decidable nonemptiness problem; the class of reduction automata and its variants [9,6,11] are a case in point. Another subclass of tree automata with equality constraints has been suggested by Bogaert and Tison [1]: they allow equality (and disequality) constraints only between sibling subtrees (i.e., direct subtrees or subtrees of depth one) and show that this class forms a Boolean algebra and that the nonemptiness problem for this class is decidable.

In this paper, we aim at extending Bogaert and Tison's automaton model to the unranked setting. However, even the definition of such automata is not obvious: with unrankedness, on the one hand, the number of pairs of sibling subtrees to be compared is not a priori bounded and may increase with the size of the input tree. On the other hand, the (possibly unboundedly many) sibling comparisons must be finitely representable in order to define an automaton model properly. Here, we propose using formulas of monadic second-order logic over the state set of the underlying automaton to address the pairs of siblings to be compared. In this way, we meet the two requirements just mentioned: unbounded number of but finitely representable equality tests between sibling subtrees.

The main result of this paper is that the nonemptiness problem for the deterministic unranked tree automata with equality and disequality constraints between siblings we propose is decidable, which we show by adapting Bogaert and Tison's nonemptiness decision procedure. As a remark, using encodings (e.g., the first-child-next-sibling encoding; cf. [16]), a standard way of transferring results from ranked trees to unranked ones, obviously fails for our purposes because the sibling relation must be destroyed by any encoding mapping unranked trees to ranked ones.

Further, regarding the use of subtrees to represent data values mentioned above, we would like to point out that, if we want to test equality only between the data values and ignore the node labels, then we actually do not want to compare whole sibling subtrees. Thus, a next step along this line of study would be to consider automaton models that do not directly compare subtrees but, instead, the output of some preprocessing of the subtrees; see Section 5 for a discussion on this.

*Outline of the paper.* After fixing our notations in Section 2, in Section 3 we present our automaton model, indicate some closure properties, and show that the nondeterministic automata are more expressive than the deterministic ones. In Section 4 we show our main result, namely that the nonemptiness problem for the deterministic case is decidable. Section 5 indicates some possible variations of our automaton model. We conclude with some remarks on further prospects in Section 6. Due to space limitations, proof details are omitted and can be found in the preliminary version of this paper [12].

*Related works.* Lugiez [14] proposes automata on multitrees (unranked, unordered trees) with a certain type of constraints among sibling multitrees in the transitions and shows that these automata are closed under Boolean operations, determinizable, and have a decidable nonemptiness problem. The constraints he uses incorporate both numerical (Presburger) constraints and inclusion relations among multisets of (multi)trees. By using Boolean combinations of constraints of the latter kind, it is then possible to impose equality tests among sibling (multi)trees, so his work also extends Bogaert and Tison's. Nevertheless, his approach is not comparable to ours in several respects. In his approach, besides unorderedness, evaluating a constraint in an unbounded (unordered) sequence of (multi)trees is reduced to evaluating the constraint in an (unordered) sequence of multisets of trees whose length is bounded by the number of states of the underlying automaton. Consequently, first, equality tests are imposed between multisets of trees (in our setting: between trees), and second, the number of equality tests depends on the number of states of the automaton instead of the size of the input (multi)tree.

## 2 Preliminaries

We denote the set of (positive) natural numbers by  $\mathbb{N}$  (respectively,  $\mathbb{N}_+$ ). For  $k > 0$ , the set of  $k$ -tuples over these sets are denoted by  $\mathbb{N}^k$  and  $\mathbb{N}_+^k$ , respectively; throughout the paper, such tuples are usually denoted by  $\vec{d}, \vec{e}, \dots$ . Further, as usual, these tuples are ordered by comparing them componentwise. Whenever  $k$  is clear from the context, we denote by  $\hat{m}$ , for  $m \in \mathbb{N}$ , the  $k$ -tuple  $(m, \dots, m)$ .

For a set  $A$ , we denote the set of all (finite) words over  $A$  by  $A^*$ . We denote the empty word by  $\varepsilon$  and write  $A^+$  for  $A^* \setminus \{\varepsilon\}$ . For a word  $w$  over  $A$ , we denote its length by  $|w|$ .

Let  $A$  be a finite, nonempty alphabet. A nonempty word  $w$  over  $A$  defines the word structure  $\langle \{1, \dots, |w|\}, S, <, (\chi_a)_{a \in A} \rangle$  where  $S$  and  $<$  denote the successor and the order relation, respectively, over the set  $\{1, \dots, |w|\}$  of positions in  $w$ , and  $\chi_a$ , for each  $a \in A$ , is the set of  $a$ -labeled positions in  $w$ . To simplify notation, we do not distinguish between a word and its corresponding word structure. The formulas of monadic second-order (MSO) logic over words over  $A$  are built up from: first-order variables  $x, y, z, \dots$ , which range over positions; monadic second-order variables  $X, Y, Z, \dots$ , which range over sets of positions; atomic formulas  $x = y$ ,  $x < y$ ,  $S(x, y)$ ,  $X(x)$ , and  $\chi_a(x)$ , for all  $a \in A$  and for all variables  $x, y, X$ ; Boolean connectives; and first-order as well as set quantifiers. We

write  $\varphi(x_1, \dots, x_n, X_1, \dots, X_m)$  to indicate that the MSO-formula  $\varphi$  may contain free occurrences of the variables  $x_1, \dots, x_n, X_1, \dots, X_m$ . If a word structure  $w$ , together with an assignment of positions  $\kappa_1, \dots, \kappa_n$  and of sets  $K_1, \dots, K_m$  of positions in  $w$  to the free variables  $x_1, \dots, x_n, X_1, \dots, X_m$ , respectively, satisfies  $\varphi$ , we write  $w \models \varphi(\kappa_1, \dots, \kappa_n, K_1, \dots, K_m)$ .

In the sequel,  $\Sigma$  will always denote a nonempty, finite (tree-labeling) alphabet. A tree domain  $D$  is a nonempty, prefix-closed subset of  $\mathbb{N}_+^*$  such that, for each  $u \in D$  and  $i > 0$ , if  $ui \in D$ , then also  $uj \in D$ , for each  $j \in \{1, \dots, i\}$ . A finite unranked tree  $t$  over  $\Sigma$  (or simply  $\Sigma$ -labeled tree in the sequel) is a mapping  $t: \text{dom}_t \rightarrow \Sigma$  where  $\text{dom}_t$  is a finite tree domain. The elements of  $\text{dom}_t$  are called the nodes of  $t$ , and the node  $\varepsilon$  is called the root of  $t$ . A node  $u \in \text{dom}_t$  is said to have  $k \geq 0$  successors if  $uk \in \text{dom}_t$  but  $u(k+1) \notin \text{dom}_t$ . In this case, we call  $ui$  the  $i$ -th successor of  $u$ , and we say that  $ui$  and  $uj$  are sibling nodes, for each  $i, j \in \{1, \dots, k\}$ . A leaf of  $t$  is a node without any successor. Given a node  $u$  of  $t$ , the subtree of  $t$  at  $u$  is the tree given by  $t|_u$  with  $\text{dom}_{t|_u} = \{v \in \mathbb{N}_+^* \mid uv \in \text{dom}_t\}$  and  $t|_u(v) = t(uv)$ , for all  $v \in \text{dom}_{t|_u}$ . Further,  $t|_u$  is called a direct subtree of  $t$  if  $|u| = 1$ . We write  $t$  as  $a(t_1 \cdots t_k)$  to indicate that its root is labeled with  $a$  and that it has  $k$  successors at which the subtrees  $t_1, \dots, t_k$  are rooted. We denote the set of all  $\Sigma$ -labeled trees by  $\mathcal{T}_\Sigma$ .

### 3 Automata with Equality and Disequality Constraints Between Siblings on Unranked Trees

In the framework of ranked trees, roughly speaking, a finite tree automaton processes a given input tree by assigning states to the nodes of the tree, say, in a bottom-up fashion, according to its transitions. The automaton model introduced in [1] extends this definition by requiring that the application of a transition on a node of the input tree is subject to some equality and disequality constraints between the direct subtrees of that particular node. For instance, “ $1 = 2 \wedge 1 \neq 3$ ” expresses the property that the first and the second subtree are equal while the first and the third subtree are different from each other. Note that the constraints directly address the subtrees to be compared, which is possible since the number of successors of any node of a ranked tree is bounded.

When moving on to the framework of unranked trees, we encounter the fact that the number of successors of a node, when applying a transition, is no longer a priori bounded by a rank. The usual approach to this phenomenon is to use regular word languages over the set of states in the transitions of a finite tree automaton instead of mere sequences of states (cf. [4]). In this way, we allow the number of successors of a node to be arbitrarily large (but finite) while ensuring a finite representation (by means of, e.g., regular expressions over the set of states) of the automaton model.

The same phenomenon occurs if we now want to add equality and disequality constraints between the direct subtrees of a node (or simply *sibling constraints* for short). As an illustration, if we want to express that “all direct subtrees are equal to one another”, then we will have to address unboundedly many pairs of

direct subtrees to be compared; in the framework of ranked trees, say, of rank  $k$ , we just need to define the constraint  $\bigwedge_{1 \leq i, j \leq k} (i = j)$ . To cope with this phenomenon, in the sequel, we will use MSO-formulas to address the pairs of direct subtrees to be compared; by doing so, we take into account the unboundedness aspect while ensuring that the constraints we use are finitely representable.

Let  $A$  be a finite, nonempty alphabet. An *atomic sibling constraint over  $A$*  is given by a pair  $(\varphi, \eta)$  where  $\varphi(x, y)$  is an MSO-formula over words over  $A$  that may contain free occurrences of the first-order variables  $x$  and  $y$ , and  $\eta$  is one of the following four types of usage:  $\exists_{\text{EQ}}$ ,  $\exists_{\text{NEQ}}$ ,  $\forall_{\text{EQ}}$ , and  $\forall_{\text{NEQ}}$ . Intuitively, an  $\exists_{\text{EQ}}$ -constraint ( $\exists_{\text{NEQ}}$ -constraint) says that “there is a pair of positions that satisfies  $\varphi$  and the subtrees at these positions are equal (or distinct, respectively)”, and a  $\forall_{\text{EQ}}$ -constraint ( $\forall_{\text{NEQ}}$ -constraint) says that “for each pair of positions that satisfies  $\varphi$  the subtrees at these positions must be equal (or distinct, respectively)”. Formally, a nonempty word  $w$  over  $A$  together with a sequence  $t_1 \dots t_{|w|}$  of  $\Sigma$ -labeled trees are said to satisfy an atomic sibling constraint  $(\varphi, \eta)$  if, depending on  $\eta$ , one of the following holds:

- $\eta = \exists_{\text{EQ}}$ : there exist  $\kappa, \lambda \in \{1, \dots, |w|\}$  such that  $w \models \varphi(\kappa, \lambda)$  and  $t_\kappa = t_\lambda$ .
- $\eta = \exists_{\text{NEQ}}$ : there exist  $\kappa, \lambda \in \{1, \dots, |w|\}$  such that  $w \models \varphi(\kappa, \lambda)$  and  $t_\kappa \neq t_\lambda$ .
- $\eta = \forall_{\text{EQ}}$ : for all  $\kappa, \lambda \in \{1, \dots, |w|\}$ , if  $w \models \varphi(\kappa, \lambda)$ , then  $t_\kappa = t_\lambda$ .
- $\eta = \forall_{\text{NEQ}}$ : for all  $\kappa, \lambda \in \{1, \dots, |w|\}$ , if  $w \models \varphi(\kappa, \lambda)$ , then  $t_\kappa \neq t_\lambda$ .

A *sibling constraint over  $A$*  is built up from atomic sibling constraints by means of Boolean connectives, and the semantics definition above is extended accordingly. The set of all sibling constraints over  $A$  is denoted by  $\text{CONS}_A$ .

We remark that  $\exists_{\text{EQ}}$ -constraints and  $\forall_{\text{NEQ}}$ -constraints are dual with respect to negation. Likewise,  $\exists_{\text{NEQ}}$ -constraints and  $\forall_{\text{EQ}}$ -constraints are dual with respect to negation. Hence, it suffices to consider only positive Boolean combinations (i.e., without negation) of atomic sibling constraints.

An *unranked tree automaton with equality and disequality constraints between siblings (UTACS)* over  $\Sigma$  is defined as a tuple  $\mathfrak{A} = (Q, \Sigma, \Lambda, \Delta, F)$  where:  $Q$  is a finite, nonempty set of states;  $F \subseteq Q$  is the set of final or accepting states;  $\Lambda \subseteq \Sigma \times Q$  contains the leaf transitions; and

$$\Delta \subseteq \text{Reg}_+(Q) \times \text{CONS}_Q \times \Sigma \times Q ,$$

where  $\text{Reg}_+(Q)$  denotes the set of regular subsets of  $Q^+$ , is the set of inner-node transitions. Given a  $\Sigma$ -labeled tree  $t$ , a run of  $\mathfrak{A}$  on  $t$  is defined as a  $Q$ -labeled tree  $\rho: \text{dom}_t \rightarrow Q$  with the following property: (a) for each leaf node  $u \in \text{dom}_t$ , we have  $(t(u), \rho(u)) \in \Lambda$ ; (b) for each node  $u \in \text{dom}_t$  with  $k \geq 1$  successors, there exists a transition  $(L, \alpha, t(u), \rho(u)) \in \Delta$  such that the word  $\rho(u_1) \dots \rho(u_k)$  belongs to  $L$  and, together with the tree sequence  $t|_{u_1} \dots t|_{u_k}$ , satisfies  $\alpha$ . In case such a run exists, we write  $t \rightarrow_{\mathfrak{A}} \rho(\varepsilon)$  or simply  $t \rightarrow \rho(\varepsilon)$ , whenever no confusion might arise, and say that  $t$  evaluates to  $\rho(\varepsilon)$ . The run  $\rho$  is said to be accepting if  $\rho(\varepsilon) \in F$ . The tree  $t$  is accepted by  $\mathfrak{A}$  if there is an accepting run of  $\mathfrak{A}$  on  $t$ . The set of trees accepted by  $\mathfrak{A}$  is denoted by  $T(\mathfrak{A})$ .

The UTACS  $\mathfrak{A}$  is called *deterministic* if, for each tree  $t \in \mathcal{T}_\Sigma$ , there exists at most one state  $q$  with  $t \rightarrow q$ .

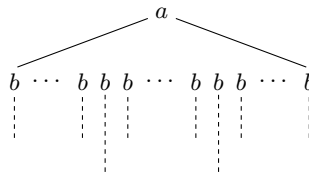


*Example 1.* The set of well-balanced trees over the alphabet  $\{a\}$  can be recognized by a UTACS by taking  $Q = F = \{q\}$ ,  $\Lambda = \{(a, q)\}$ , and  $\Delta = \{(Q^+, \alpha, a, q)\}$  with  $\alpha = (\text{true}, \forall_{\text{EQ}})$ .

By adapting the standard constructions from the ranked setting (see, for example, [7]), one can show that the class of (nondeterministic) automata with constraints between siblings on unranked trees is closed under union and intersection, and that the class of deterministic automata is closed under complementation. On the other hand, the nondeterministic automata, as opposed to the ranked case, are more powerful than the deterministic ones (see Proposition 1 below); this fact, in turn, raises the question whether the class of nondeterministic UTACS's is closed under complementation.

**Proposition 1.** *There exists a tree language that is recognizable by a nondeterministic UTACS, but by no deterministic UTACS.*

The tree language that we use to separate the two classes consists of trees of the form depicted in Figure 1. Intuitively, such a tree consists of a root labeled



**Fig. 1.** Trees separating nondeterministic and deterministic UTACS's (the dashed lines represent  $b$ -strands)

with  $a$  and below it strands of  $b$ 's. All but two of the  $b$ -strands are of the same length, and the two special  $b$ -strands themselves are of the same length. With nondeterminism, essentially, we would guess the positions of the latter  $b$ -strands and mark them by means of a special state. Then, using this particular state, we can address the appropriate pairs of positions that should be equal and those that should be distinct. With determinism, this is no longer possible; the fact that there are  $b$ -strands of arbitrary length prevents the possibility of using a special state to mark the positions of the two special  $b$ -strands and thus also of addressing their positions in the constraints.

## 4 Nonemptiness Problem: The Deterministic Case

In the ranked setting, it has been shown in [1] that the nonemptiness problem for deterministic automata with sibling constraints is decidable, which carries over into nondeterministic automata since the latter can be determinized. The method used there is an adaptation of the standard marking algorithm: one constructs trees that are accepted by the automaton under consideration, in a bottom-up fashion, by applying the transitions of the automaton. In order

to apply a transition, in turn, one needs to find, for each state occurring in the transition, a tree that evaluates to this state. With disequality constraints, however, we may need more than one tree evaluating to a state; for instance, if a transition requires that the first and the second subtree both evaluate to the same state, say  $p$ , and that they are distinct, then at least two distinct trees evaluating to  $p$  are needed to apply the transition. Of course, if we want to apply a transition to a node, then we need, for each state occurring in the transition, at most only as many distinct trees as the number of successors of this node. Thus, if the number of successors a tree node may have is bounded, then this bound gives an upper bound on the sufficient number of distinct trees needed for each state in order to apply a transition.

Now, our main obstacle to transferring the above nonemptiness decision procedure to the unranked setting indeed lies in the unrankedness aspect: as the number of successors of a tree node is not a priori bounded, we first need to find out how we can bound the sufficient number of distinct trees needed to satisfy a sibling constraint. In Lemma [11](#) below, we assert the existence of such a bound: for each transition, if this transition is applicable, then as many distinct trees as given by this bound are sufficient in order to apply this transition. Using this bound, we can then devise, based on the corresponding algorithm in the ranked setting, a nonemptiness decision procedure for deterministic UTACS's.

In the remainder of this section, unless stated otherwise, let  $\mathfrak{A} = (Q, \Sigma, \Delta, \Lambda, F)$  be a deterministic UTACS and  $\tau = (L, \alpha, a, q)$  be a transition of  $\mathfrak{A}$ .

We call a word  $w \in Q^+$  *suitable for  $\tau$*  if it can be used in an application of  $\tau$ , thus resulting in a tree that evaluates to  $q$ , provided that, for each state occurring in  $w$ , there are plenty of (i.e., sufficiently many) distinct trees evaluating to this state. Note that not every word is suitable for  $\tau$ ; for instance, if  $\alpha$  requires that the subtrees at the first and the second position of  $w$  are equal, then the states at these positions must be the same, too, since  $\mathfrak{A}$  is deterministic. Let us denote the set of words that are suitable for  $\tau$  by  $S_\tau$ . Now, in order to analyze the applicability of  $\tau$ , it suffices only to consider words that are suitable for  $\tau$ , for if a word  $w$  is not suitable for  $\tau$ , then there is no sequence of trees together with which  $w$  can both belong to  $L$  and satisfy the constraint  $\alpha$ . Moreover, in the following exposition we can assume that  $S_\tau$  is not empty as  $\tau$  otherwise cannot be applied at all and can thus be removed from  $\Delta$ .

For a  $\tau$ -suitable word  $w$ , let  $\llbracket w, \tau \rrbracket \in \mathbb{N}^{|Q|}$  be a tuple of natural numbers that indicates, for each state, the number of distinct trees that are used for a particular application of  $\tau$  that uses  $w$ . Alternatively,  $\llbracket w, \tau \rrbracket$  can be seen as a mapping  $\llbracket w, \tau \rrbracket: Q \rightarrow \mathbb{N}$  where  $\llbracket w, \tau \rrbracket(p)$  is assigned the  $p$ -component of  $\llbracket w, \tau \rrbracket$ , for each  $p \in Q$ . We would like to point out that the value of  $\llbracket w, \tau \rrbracket$  does not solely depend on  $w$  and  $\tau$ , but also on a certain application of  $\tau$  by means of  $w$ . In order to simplify our presentation, whenever we pick a  $\tau$ -suitable word  $w$ , in the following, we always implicitly refer to such a particular application of  $\tau$ , which then gives a unique value of  $\llbracket w, \tau \rrbracket$ . We remark that, in general,  $\llbracket w, \tau \rrbracket(p)$ , for each  $p \in Q$ , does not need to exceed  $|w|$ .

Our aim is to show the existence of a bound  $N$  such that for each word  $w$  that is suitable for  $\tau$ , if  $\llbracket w, \tau \rrbracket(p)$  exceeds  $N$ , for some  $p \in Q$ , then we can find another  $\tau$ -suitable word  $w'$  such that  $\llbracket w', \tau \rrbracket$  is less than or equal to  $\widehat{N}$ . This is stated in the following lemma, to which we will refer to as the bound lemma.

**Lemma 1.** *There exists some  $N \geq 0$  such that, for each transition  $\tau$  of  $\mathfrak{A}$  and for each word  $w \in S_\tau$ , there exists a word  $w' \in S_\tau$  such that the following holds:*

$$\llbracket w', \tau \rrbracket \leq \widehat{N} \tag{1}$$

$$\llbracket w', \tau \rrbracket \leq \llbracket w, \tau \rrbracket \tag{2}$$

$$\text{For any } p \in Q, \text{ if } \llbracket w, \tau \rrbracket(p) > N, \text{ then } \llbracket w', \tau \rrbracket(p) > 0. \tag{3}$$

In essence, the lemma asserts that, if a transition  $\tau$  can be applied by means of the word  $w$ , then we can replace  $w$  with another word  $w'$  such that, for each state  $p \in Q$ , the sufficient number of distinct trees evaluating to  $p$  that are needed to apply  $\tau$  by means of  $w'$  exceeds neither  $N$  nor the corresponding number when  $w$  is used instead of  $w'$ . The third condition in the bound lemma is needed for technical reasons; it asserts that if a component  $p \in Q$  in  $\llbracket w, \tau \rrbracket$  exceeds  $N$ , then it must occur in  $w'$ .

Before we sketch our method of finding such a bound, let us first introduce some further notations. We recall that a word  $w \in Q^+$  is suitable for  $\tau$  if, given plenty of distinct trees for each state occurring in  $w$ , the transition  $\tau$  can be applied. Now, given a set  $R \subseteq Q$  and a tuple  $\bar{d} \in \mathbb{N}^{|R|}$ , the word  $w$  is said to be *suitable for  $\tau$  with respect to  $R$  and  $\bar{d}$*  if the transition  $\tau$  can be applied under the assumption that for each state  $p$  occurring in  $w$ : (a) there are  $\bar{d}(p)$  many distinct trees that evaluate to  $p$ , if  $p \in R$ , and (b) there are plenty of (i.e., sufficiently many) distinct trees that evaluate to  $p$ , if  $p \notin R$ . We denote the set of all words that are suitable for  $\tau$  with respect to  $R$  and  $\bar{d}$  by  $S_{\tau,R,\bar{d}}$ .

**Lemma 2.** *The sets  $S_\tau$  and  $S_{\tau,R,\bar{d}}$ , for all  $R \subseteq Q$  and  $\bar{d} \in \mathbb{N}^{|R|}$ , are regular subsets of  $Q^+$ . In particular, the nonemptiness problem for these sets is decidable.*

*Proof (sketch).* Roughly speaking, a word  $w$  belongs to  $S_\tau$  iff it belongs to  $L$  and the constraints in  $\alpha$  do not cause conflicts in  $w$ ; for instance, any pair  $(\kappa, \lambda)$  of positions in  $w$  satisfying a  $\forall_{\text{EQ}}$ -constraint of  $\tau$  may not satisfy any  $\forall_{\text{NEQ}}$ -constraint of  $\tau$ . In addition, since  $\mathfrak{A}$  is supposed to be deterministic, if a pair of positions is declared to have equal subtrees by  $\alpha$ , then the  $Q$ -labels of those positions must be equal. Since atomic constraints are built from MSO-formulas, we can write an MSO-formula that captures all these requirements, which shows the regularity of  $S_\tau$ . To show the regularity of  $S_{\tau,R,\bar{d}}$ , we just need to additionally require that the occurrences of  $p \in R$  can be partitioned into  $\bar{d}(p)$ -many sets of positions such that this partitioning does not cause conflict in  $w$ ; for instance, if a pair  $(\kappa, \lambda)$  of positions in  $w$  that are labeled with a state from  $R$  satisfies a  $\forall_{\text{EQ}}$ -constraint, then both positions must lie in the same partition.  $\square$

Let us now illustrate our method of finding the desired bound for a fixed transition  $\tau$  by means of a simple example. Suppose  $q_1, \dots, q_4$  are the states of  $\mathfrak{A}$

and  $q_1$  is the target state of  $\tau$ , and suppose  $v \in S_\tau$ , say, with  $\llbracket v, \tau \rrbracket = (2, 4, 5, 1)$ . Then,  $\llbracket v, \tau \rrbracket$  already gives a first approximation of the bound, namely 5. That is, if, for each state, there are five distinct trees that evaluate to this state, then we can apply  $\tau$  (using  $v$ ). Now, what happens if there are actually, say, only one tree for  $q_1$  and three distinct trees for  $q_2$ ? Then, two cases may occur. First, it might be the case that  $\tau$  cannot be applied at all, i.e., it is not possible to apply  $\tau$  under these conditions (with only one tree for  $q_1$  and three distinct trees for  $q_2$ ). Otherwise, second, there is an application of  $\tau$  under these conditions, for example, using a word  $v'$ , six distinct trees for  $q_3$ , and seven distinct trees for  $q_4$ . In the latter case, the bound must then be updated to 7 in order to make certain that we do not miss out any possibility of applying  $\tau$ .

Recapitulating, we proceed for a fixed transition  $\tau$  as follows:

- (i) Start with an initial bound  $N_\tau$ .
- (ii) Check, for all subsets  $R$  of  $Q$  and all tuples  $\bar{d} \in \mathbb{N}^{|R|}$  with  $\bar{d} \leq \widehat{N}_\tau$ , whether the set  $S_{\tau, R, \bar{d}}$  is nonempty (which is possible due to Lemma 2).
- (iii) In this case, pick a word  $v_{\tau, R, \bar{d}}$ , which then gives a new approximation of the desired bound via  $\llbracket v_{\tau, R, \bar{d}}, \tau \rrbracket$ , and update  $N_\tau$  accordingly.
- (iv) Go back to (ii).

Upon termination of this procedure, which turns out to be non-trivial and relies on Dickson’s Lemma 10 (see also 5, Lemma 3), the value of  $N_\tau$  gives the desired bound with respect to the particular transition  $\tau$ . For the bound required by Lemma 1, we then take the maximum of the bounds  $N_\tau$  among all the transitions  $\tau$  of  $\mathfrak{A}$ . For more details, in particular, concerning the termination and the correctness of the procedure, the reader is referred to 12, Section 4.3].

We now present an algorithm that, given a deterministic UTACS, decides whether the corresponding tree language is nonempty. In essence, this algorithm is an adaptation of the standard marking algorithm: it consists of a main loop that in each round collects, for each state, a tree resulting from the application of a transition based on the trees collected from the previous rounds. The bound from Lemma 1 gives the sufficient number of distinct trees that we ought to collect for each state; the main loop is iterated until either we cannot construct new trees anymore, or we have collected, for each state, as many trees as the bound. Hence, the algorithm eventually terminates.

**Algorithm 1.** Given a deterministic UTACS  $\mathfrak{A} = (Q, \Sigma, \Lambda, \Delta, F)$ , together with the bound  $N$  from Lemma 1, the algorithm NONEMPTY( $\mathfrak{A}$ ) decides whether  $T(\mathfrak{A}) \neq \emptyset$  holds. The tuple  $\bar{d} \in \mathbb{N}^{|Q|}$  keeps track of the number of trees we have collected so far; for each state  $q \in Q$ , we use  $T_q$  to store the trees evaluating to  $q$ , so at any time of the computation  $\bar{d}(q)$  contains the current value of  $|T_q|$ .

- ```

1: function NONEMPTY( $\mathfrak{A}$ )
2:   initialize each  $T_q$  with  $\{a \in \Sigma \mid (a, q) \in \Lambda\}$ 
3:   repeat
4:     if there exist  $\tau = (L, \alpha, a, q) \in \Delta$ ,  $w = q_1 \dots q_m \in S_{\tau, Q, \bar{d}}$ , and  $t_1 \in T_{q_1}$ ,
        $\dots, t_m \in T_{q_m}$  such that  $w$  and  $t_1 \dots t_m$  satisfy  $\alpha$ , and  $|T_q| < N$ 
5:     then  $T_q \leftarrow T_q \cup \{a(t_1 \dots t_m)\}$ 

```

- 6: **until** no new tree can be constructed, or we have  $\bar{d} = \widehat{N}$   
 7: **if**  $T_q \neq \emptyset$  for some  $q \in F$  **then return** ' $T(\mathfrak{A}) \neq \emptyset$ '  
 8: **else return** ' $T(\mathfrak{A}) = \emptyset$ '

The algorithm is sound since in Line 3-6 trees are constructed according to the transition relation  $\Delta$  of  $\mathfrak{A}$ . The completeness of the algorithm (i.e., if  $T(\mathfrak{A})$  is nonempty, then its output must be ' $T(\mathfrak{A}) \neq \emptyset$ ') follows immediately from Lemma 3 below, which asserts that, if a tree  $t$  evaluates to a state  $q$ , then either we will eventually construct it, or we have already had  $N$  trees evaluating to  $q$ .

**Lemma 3.** *For any  $t \in \mathcal{T}_\Sigma$  and  $q \in Q$ , if  $t \rightarrow q$ , then  $t \in T_q$  or  $|T_q| = N$  holds.*

We prove this lemma by an induction on the structure of  $t$  (cf. [12, Lemma 14]). The more interesting case is embodied by the induction step, i.e., the case  $t = a(t_1 \dots t_m)$  with  $t \rightarrow q$  and  $t_i \rightarrow q_i$ , for each  $i = 1, \dots, m$ , where  $t$  itself does not belong to  $T_q$ . Then, we have to construct  $N$  distinct trees evaluating to  $q$  out of the trees in  $\bigcup_{s \in Q} T_s$ . For this, we might have to replace the word  $q_1 \dots q_m$  with one that satisfies the requirements of Lemma 1 in order to obtain trees evaluating to  $q$  that are actually constructed by Algorithm 1. In particular, the requirement (3) in the bound lemma ensures that we are indeed able to construct  $N$  such trees. To sum up, we obtain the main result of this section:

**Theorem 2.** *The nonemptiness problem for deterministic UTACS's is decidable.*

Throughout our algorithms and proofs, actually, we have just used the regularity of the sets of suitable words without really analyzing the form of the equality and disequality constraints appearing in the transitions at all. Hence, our method will still work if we vary the definition of the constraints, as long as the regularity of the sets of suitable words or, more generally, the decidability of the nonemptiness problem for these sets, is maintained.

As a remark, our method does not work for nondeterministic UTACS's. With determinism, we can assume that, if two trees evaluate to two different states, then these trees must be different as well; this property fails to hold for nondeterministic UTACS's. In fact, this observation has then lead us to define the notion of suitability of words over the set of states with respect to a transition, thereby reducing the analysis of the distinctness among trees to that among states.

## 5 Restrictions and Extensions of the Model

In this section, we indicate some possible variations of the automaton model we have introduced and discuss how far our results, in particular with respect to the decidability of the nonemptiness problem, are retained.

*Sibling constraints without references to states.* We recall that the atomic constraints between siblings we have used in the definition of UTACS are given by MSO-formulas over the state set of the underlying UTACS. In other words, the MSO-formulas used as constraints may refer to states when defining the pairs of sibling subtrees that are supposed to be equal or distinct. In fact, the use of this

ability has been demonstrated in Proposition 11 to show that the tree language given there is recognizable by a nondeterministic UTACS.

With this phenomenon in mind, we now prohibit the reference to states in the atomic constraints: the MSO-formulas used as atomic constraints now lack atomic MSO-formulas of the form  $\chi_a(x)$ . With this definition, we can then show that every nondeterministic restricted UTACS can be transformed into a deterministic one by using the standard subset construction.

*Comparing the output of a tree transducer.* When applying a transition, given a pair of siblings to be compared, what we do up to now is to check whether the subtrees below these positions are equal or not. A more involved processing is to feed the subtrees into a (deterministic) tree transducer and compare the output trees instead of the subtrees themselves, for example, if we want to compare only data values contained in the trees instead of the whole trees, as indicated in Section 11. Furthermore, if we consider bottom-up tree transducers, we can use MSO-formulas over the state set of the transducer instead of the state set of the underlying UTACS.

However, if we want to apply our method to solve the nonemptiness problem for the resulting automaton model, similar to our remark at the end of Section 4, we must require that if the states assumed by the transducer under consideration after producing two output trees, say,  $t$  and  $t'$ , are different, then  $t$  and  $t'$  must also be different. Hence, a more restricted model of tree transducers, which are, in some sense, output deterministic, is required.

## 6 Conclusions

We have extended the tree automaton model defined by Bogaert and Tison in 11 to the case of unranked trees. In the transitions, we use MSO-formulas to address the pairs of positions to be compared. It then turns out that the nondeterministic model, in contrast to the ranked setting, is more expressive than the deterministic one. Our main result is that the nonemptiness problem for the latter model is decidable: we adapt the standard marking algorithm, which collects for each state a sufficient number of distinct trees. For this number, we define an appropriate bound by means of Lemma 11.

As far as the complexity of our nonemptiness decision procedure is concerned, there are two issues that still need to be settled. First, the termination of the bound algorithm is given by Dickson's Lemma, so its time complexity depends on the length of a certain antichain. Second, the complexity of our algorithms depends on the representation of the MSO-formulas that are used as constraints. It is known, however, that the translation from MSO-formulas to automata on words might involve a non-elementary blow-up. Thus, a careful choice of the representation of the sibling constraints is needed in order to analyze the complexity of our algorithms.

In addition to analyzing complexity, other prospective future work includes: (a) studying the nonemptiness problem for nondeterministic UTACS, (b) considering extensions that involve tree transducers, as indicated in Section 5.

(c) considering automaton models that allow determinization, and (d) studying the connection to automata on words and trees with data considered in [32].

## References

1. Bogaert, B., Tison, S.: Equality and disequality constraints on direct subterms in tree automata. In: Finkel, A., Jantzen, M. (eds.) STACS 1992. LNCS, vol. 577, pp. 161–171. Springer, Heidelberg (1992)
2. Bojańczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data trees and XML reasoning. In: PODS 2006, pp. 10–19. ACM Press, New York (2006)
3. Bojańczyk, M., Muscholl, A., Schwentick, T., Segoufin, L., David, C.: Two-variable logic on words with data. In: LICS 2006, pp. 7–16. IEEE Computer Society Press, Los Alamitos (2006)
4. Brüggemann-Klein, A., Murata, M., Wood, D.: Regular tree and regular hedge languages over unranked alphabets. Research Report HKUST-TCSC-2001-05, HKUST Theoretical Computer Science Center (2001), Available on <http://hdl.handle.net/1783.1/738>
5. Burkart, O., Caucal, D., Moller, F., Steffen, B.: Verification on infinite structures. In: Handbook of Process Algebra, pp. 545–623. Elsevier, Amsterdam (2001)
6. Caron, A.C., Comon, H., Coquidé, J.L., Dauchet, M., Jacquemard, F.: Pumping, cleaning and symbolic constraints solving. In: Shamir, E., Abiteboul, S. (eds.) ICALP 1994. LNCS, vol. 820, pp. 436–449. Springer, Heidelberg (1994)
7. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications. (1997) Current release: October 1st, 2002 Available on <http://www.grappa.univ-lille3.fr/tata>
8. Dal Zilio, S., Lugiez, D.: XML schema, tree logic and sheaves automata. In: Nieuwenhuis, R. (ed.) RTA 2003. LNCS, vol. 2706, pp. 246–263. Springer, Heidelberg (2003)
9. Dauchet, M., Caron, A.C., Coquidé, J.L.: Automata for reduction properties solving. *Journal of Symbolic Computation* 20, 215–233 (1995)
10. Dickson, L.E.: Finiteness of the odd perfect and primitive abundant numbers with  $n$  distinct prime factors. *American Journal of Mathematics* 35, 413–422 (1913)
11. Jacquemard, F., Rusinowitch, M., Vigneron, L.: Tree automata with equality constraints modulo equational theories. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 557–571. Springer, Heidelberg (2006)
12. Krianto, W., Löding, C.: Unranked tree automata with sibling equalities and disequalities. Technical Report AIB-2006-13, RWTH Aachen (2006)
13. Libkin, L.: Logics for unranked trees: An overview. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 35–50. Springer, Heidelberg (2005)
14. Lugiez, D.: Counting and equality constraints for multitree automata. In: Gordon, A.D. (ed.) ETAPS 2003 and FOSSACS 2003. LNCS, vol. 2620, pp. 328–342. Springer, Heidelberg (2003)
15. Mongy-Steen, J.: Transformation de noyaux reconnaissables d’arbres. Forêts RATEG. PhD thesis, Université de Lille I (1981)

16. Neven, F.: Automata, logic, and XML. In: Bradfield, J.C. (ed.) CSL 2002 and EACSL 2002. LNCS, vol. 2471, pp. 2–26. Springer, Heidelberg (2002)
17. Seidl, H., Schwentick, T., Muscholl, A.: Numerical document queries. In: PODS 2003, pp. 155–166. ACM Press, New York (2003)
18. Seidl, H., Schwentick, T., Muscholl, A., Habermehl, P.: Counting in trees for free. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 1136–1149. Springer, Heidelberg (2004)
19. Tommasi, M.: Automates d'arbres avec tests d'égalité entre cousins germains. Technical report, Mémoire de DEA, Université de Lille I (1992)



# Regular Languages of Nested Words: Fixed Points, Automata, and Synchronization

Marcelo Arenas<sup>1</sup>, Pablo Barceló<sup>2</sup>, and Leonid Libkin<sup>3</sup>

<sup>1</sup> Pontificia Universidad Católica de Chile

<sup>2</sup> Universidad de Chile

<sup>3</sup> University of Edinburgh

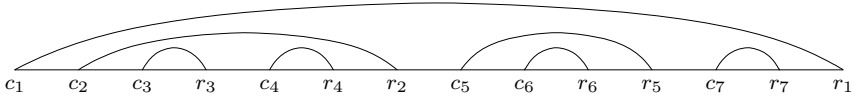
**Abstract.** Nested words are a restriction of the class of visibly pushdown languages that provide a natural model of runs of programs with recursive procedure calls. The usual connection between monadic second-order logic (MSO) and automata extends from words to nested words and gives us a natural notion of regular languages of nested words.

In this paper we look at some well-known aspects of regular languages – their characterization via fixed points, deterministic and alternating automata for them, and synchronization for defining regular relations – and extend them to nested words. We show that mu-calculus is as expressive as MSO over finite and infinite nested words, and the equivalence holds, more generally, for mu-calculus with past modalities evaluated in arbitrary positions in a word, not only in the first position. We introduce the notion of alternating automata for nested words, show that they are as expressive as the usual automata, and also prove that Muller automata can be determinized (unlike in the case of visibly pushdown languages). Finally we look at synchronization over nested words. We show that the usual letter-to-letter synchronization is completely incompatible with nested words (in the sense that even the weakest form of it leads to an undecidable formalism) and present an alternative form of synchronization that gives us decidable notions of regular relations.

## 1 Introduction

The class of *visibly pushdown languages* (VPL) has been introduced by Alur and Madhusudan [5] as a restriction of the class of context-free languages that subsumes all regular properties and some non-regular properties relevant in program analysis (e.g. stack-inspection properties and pre-post conditions). VPLs in many ways resemble regular languages: they have the same closure properties, and most natural problems related to them are decidable. The intuitive idea of VPLs is that the input alphabet  $\Sigma$  is partitioned into three parts,  $\Sigma_c, \Sigma_r, \Sigma_i$ , of symbols viewed as procedure calls, returns, and internal operations. A machine model for VPLs is a special pushdown automaton that pushes a symbol onto the stack in a call, pops one symbol in a return, and does not touch the stack when reading an internal symbol.

*Nested words* [6] replaced the implicit nesting structure of calls and returns by an explicit relation that matches calls and returns. A nested word is thus a word extended with a hierarchical structure on top of its linear structure. An example of such a nested structure of matching calls  $c_i$  and returns  $r_i$  is given below.



Such structures naturally appear, for instance, in XML documents that are string representations of trees using opening and closing tags [23,8], or in software verification of programs with stack-based control flow [4,2]. A *nested word automaton* [6] runs from left to right, similarly to a finite state automaton, but each time it encounters a “return” position, the next state depends not only on the current state but also on the state of the matching “call”.

A nice property of nested words and their automata is that they share logical characterizations with the usual (unnested) words: the automaton model has the same expressiveness as monadic second-order logic (MSO) [5,6]. This gives us a natural and robust notion of *regular languages* of nested words, with the expected closure properties, decision procedures, and logical characterizations.

For finite or infinite unnested words, an alternative way of describing regularity logically is via the modal  $\mu$ -calculus (cf. [7]). That is,  $\mu$ -calculus formulae evaluated in the first position of a word define precisely the regular languages. Moreover,  $\mu$ -calculus formulae with past modalities evaluated in an arbitrary position of a word have precisely the power of MSO formulae with one free first-order variable. As our first result, we extend these equivalences to the case of finite and infinite nested words.

We then look at automata characterizations of VPLs and nested words. Nondeterministic and deterministic automata have previously been considered [5,6,18], and [5] showed that automata can be determinized in the finite case, but in the infinite case this is impossible even for automata with a Muller acceptance condition (unlike in the case of the usual  $\omega$ -words), if one considers VPLs. Then [18] introduced a different automaton model and showed that it admits a determinization procedure over nested words. We expand this in two ways. First we introduce alternation in the case of nested word automata, and prove that alternating automata can still be translated into nondeterministic ones. Second, we refine the determinization procedure for automata from [18] to show that over infinite nested words, every regular language is definable by a deterministic Muller automaton. This also gives us some corollaries about the structure of regular languages of nested  $\omega$ -words.

We finally turn our attention to the notion of regular *relations*. Over words, one moves from sets to relations by using letter-to-letter synchronization. That is, an automaton runs over a tuple of words viewing the tuple of  $i$ th letters of the words as a single letter of an expanded alphabet [15]. The same approach works for trees, ranked and unranked [11]. The notion of regular relations also leads to a notion of automatic structures [12,13,10], i.e. decidable first-order structures over words in which all definable relations are regular.

Here we show that, in contrast to the case of words and trees, the notion of letter-to-letter synchronization is incompatible with nested words: the simplest extension of nested word automata with such synchronization is undecidable. We present an alternative call-return notion of synchronization, and show that it gives us a natural concept of regular relations over nested words.

**Related work.** VPLs were introduced in [5] and nested words in [6]. They can be viewed as special classes of trees (and we shall use this often in the paper); such tree representations were introduced in [5,6] as well. Applications in program analysis are discussed, e.g., in [2,4], and applications in processing tree-structured data in [23,8].

There are several related results on  $\mu$ -calculus and MSO, e.g. their equality over infinite binary trees [20] or finite unranked trees [9] or expressive-completeness of  $\mu$ -calculus [16]. We explain in Section 3 why we cannot derive our result from those. Another fixed-point logic  $VP_\mu$  is defined in [2] to specify properties of executions of programs. It differs from the standard versions of  $\mu$ -calculus we look at as its fixed points are evaluated not over sets of nodes but over sets of subtrees of the program; further, its expressiveness is known to be different from MSO [3].

Automata for VPLs and nested words were defined in [5,6], and [5] observed that Muller automata are not determinizable. Then [18] noticed that this is due to VPLs having potentially arbitrarily many unmatched calls/returns, and introduced a different automaton model (stair automata) that can be determinized. We use them to show how to determinize Muller automata over nested  $\omega$ -words. None of these papers addresses alternating automata over nested words.

Letter-to-letter synchronization for defining regular relations is an old notion [15], and the concept of universal automatic structures [13,12] is based on it. Although such automatic structures exist for both words and trees [10,11], we show here that letter-to-letter synchronization is incompatible with nesting structure.

**Organization.** Basic definitions are given in Section 2. We describe MSO unary queries via  $\mu$ -calculus in Section 3. In Section 4 we study automata for nested words, define alternating automata, and describe determinization for Muller automata. In Section 5 we look at synchronization and regular relations for nested words.

## 2 Preliminaries

**Words,  $\omega$ -words, and automata.** Let  $\Sigma$  be a finite alphabet. A finite word  $w = a_1 \dots a_n$  in  $\Sigma^*$  is represented as a logical structure  $\langle \{1, \dots, n\}, (P_a)_{a \in \Sigma}, < \rangle$ , where  $<$  is the usual linear order on  $\{1, \dots, n\}$ , and  $P_a$  is the set of  $i$ 's such that  $a_i = a$ . We shall use  $w$  to refer to both the word and its logical representation. Infinite, or  $\omega$ -words, are sequences  $a_1 a_2 \dots$  of symbols in  $\Sigma$  indexed by positive natural numbers, and are represented as structures  $\langle \mathbb{N}^+, (P_a)_{a \in \Sigma}, < \rangle$ . The length of  $w$  is denoted by  $|w|$ .

A (nondeterministic) *automaton*  $\mathcal{A}$  over  $\Sigma$  is a tuple  $(\Sigma, Q, Q_0, \delta, F)$ , where  $Q$  is a finite set of states,  $Q_0 \subseteq Q$  is a set of initial states,  $F \subseteq Q$  is a set of final states and  $\delta : Q \times \Sigma \rightarrow 2^Q$  is a transition function. For automata over  $\omega$ -words we shall use either a Büchi acceptance condition (given by  $F \subseteq Q$ ) or a Muller acceptance condition (given by  $\mathcal{F} \subseteq 2^Q$ ). A *run* of  $\mathcal{A}$  over a word  $w$  is a map  $\rho : \{1, \dots\} \rightarrow Q$  such that  $\rho(1) \in Q_0$  and  $\rho(i+1) \in \delta(\rho(i), a_i)$ , for all  $i$ . A finite run is accepting if  $\rho(|w|+1) \in F$ . We let  $Inf(\rho)$  be the set of states that occurs infinitely often in an infinite run  $\rho$ . Then  $\rho$  is accepting for a Büchi condition  $F$  if  $Inf(\rho) \cap F \neq \emptyset$ , and it is accepting for a Muller condition  $\mathcal{F}$  if  $Inf(\rho) \in \mathcal{F}$ . A word is accepted iff there exists an accepting run. Sets of ( $\omega$ -)words accepted by automata are called *regular*.

$\mathcal{A}$  is *deterministic* if  $|Q_0| = 1$ , and  $|\delta(q, a)| = 1$  for every  $a \in \Sigma$  and  $q \in Q$ . Nondeterministic automata over  $\omega$ -words with Büchi and Muller conditions are equivalent, and automata with Muller acceptance condition can be determinized, cf. [25].

**Nested words.** A finite *nested word* over  $\Sigma$  is a pair  $\bar{w} = (w, \eta)$ , where  $w \in \Sigma^*$  and  $\eta$  is a binary *matching relation* on  $\{1, \dots, |w|\}$  that satisfies: (1)  $\eta(i, j)$  implies  $i < j$ ; (2)  $\eta(i, j)$  and  $\eta(i, j')$  imply  $j = j'$  and  $\eta(i, j)$  and  $\eta(i', j)$  imply  $i = i'$ ; and (3) if  $\eta(i, j)$ ,  $\eta(i', j')$ , and  $i < i'$  then either  $j < j'$  or  $j' < j$ . A *nested  $\omega$ -word* is a pair  $\bar{w} = (w, \eta)$ , where  $w$  is an  $\omega$ -word and  $\eta$  is a matching on  $\mathbb{N}^+$ . We also refer to them as infinite nested words. We represent nested words as logical structures over the vocabulary  $\{(P_a)_{a \in \Sigma}, <, \eta\}$ , i.e. words expanded with a matching relation. For a nested word  $\bar{w}$  and two positions  $i < j$ , we let  $\bar{w}[i, j]$  be the substructure of  $\bar{w}$  induced by positions  $\ell$  such that  $i \leq \ell \leq j$ . A position  $i$  of a nested word  $\bar{w}$  is: (1) a *call* position if there is  $j$  such that  $\eta(i, j)$  holds; (2) a *return* position if there is  $j$  such that  $\eta(j, i)$  holds; and (3) an *internal* position if it is neither a call nor a return. Whenever  $\eta(i, j)$  holds we say that  $i$  is the call of  $j$ , and  $j$  is the return of  $i$ .

**Nested word automata.** A *nested word automaton*, or NWA [6],  $\mathcal{A}$  over  $\Sigma$  is defined as a usual automaton, except that  $\delta$  is a triple  $(\delta_c, \delta_l, \delta_r)$  of transition functions  $\delta_c, \delta_l : Q \times \Sigma \rightarrow 2^Q$ , and  $\delta_r : Q \times Q \times \Sigma \rightarrow 2^Q$ . A *run* of  $\mathcal{A}$  over  $\bar{w} = (a_1 \dots, \eta)$  is a mapping  $\rho : \{1, \dots\} \rightarrow Q$  such that  $\rho(1) \in Q_0$  and for every  $i \in \mathbb{N}^+$  (or  $i \in [1, |\bar{w}|]$  for finite nested words),

- if  $i$  is a call position, then  $\rho(i + 1) \in \delta_c(\rho(i), a_i)$ ;
- if  $i$  is an internal position, then  $\rho(i + 1) \in \delta_l(\rho(i), a_i)$ ;
- if  $i$  is a return position whose call is  $j$ , then  $\rho(i + 1) \in \delta_r(\rho(i), \rho(j), a_i)$ .

Büchi and Muller acceptance conditions can then be defined in exactly the same way as for the usual automata (and are easily shown to be equivalent over nested words, for nondeterministic automata). We refer to such automata as  $\omega$ -NWAs. An NWA is deterministic if the values of all transition functions are singletons.

A class of nested ( $\omega$ -)words accepted by an ( $\omega$ -)NWA is called *regular*.

**Monadic second-order logic and  $\mu$ -calculus.** Monadic second-order logic (MSO) extends first-order logic with quantification over sets. Over nested words, its vocabulary contains predicates  $P_a$  ( $a \in \Sigma$ ),  $<$  and  $\eta$ . A class of nested ( $\omega$ -)words is regular iff it is definable by an MSO sentence [5][6].

The  $\mu$ -calculus over nested words, denoted by  $L_\mu$ , is defined by the grammar:

$$\varphi, \varphi' := a \mid X \mid \varphi \vee \varphi' \mid \varphi \wedge \varphi' \mid \neg \varphi \mid \diamond \varphi \mid \diamond_\eta \varphi \mid \mu X. \varphi(X)$$

with  $X$  occurring positively in  $\varphi(X)$ , and  $a \in \Sigma \cup \{\text{call}, \text{int}, \text{ret}\}$ . Given a nested ( $\omega$ -)word  $\bar{w}$ , a position  $i$  in  $\bar{w}$ , and a valuation  $v$  assigning each free variable  $X$  a set of positions of  $\bar{w}$ , the semantics is as follows (omitting the rules for Boolean connectives):

- $(\bar{w}, v, i) \models \text{int}$  iff  $i$  is an internal position;  $(\bar{w}, v, i) \models \text{call}$  iff  $i$  is a call position; and  $(\bar{w}, v, i) \models \text{ret}$  iff  $i$  is a return position.
- $(\bar{w}, v, i) \models a$ , for  $a \in \Sigma$ , iff  $i$  is labeled  $a$ .
- $(\bar{w}, v, i) \models X$  iff  $i \in v(X)$ .

- $(\bar{w}, v, i) \models \diamond\varphi$  iff  $i + 1$  belongs to  $\bar{w}$  and  $(\bar{w}, v, i + 1) \models \varphi$ .
- $(\bar{w}, v, i) \models \diamond_{\eta}\varphi$  iff there is an  $\ell$  such that  $\eta(i, \ell)$  holds and  $(\bar{w}, v, \ell) \models \varphi$ .
- $(\bar{w}, v, i) \models \mu X.\varphi(X)$  iff  $i$  is in the least fixed point of the operator defined by  $\varphi$ ; in other words, if  $i \in \bigcap \{P \mid \{i' \mid (\bar{w}, v[P/X], i') \models \varphi\} \subseteq P\}$ , where  $v[P/X]$  extends the valuation  $v$  with  $v(X) = P$ .

The  $\mu$ -calculus over words does not mention the modality  $\diamond_{\eta}\varphi$ .

We shall also work with the *full*  $\mu$ -calculus [28] (denoted by  $L_{\mu}^{\text{full}}$ ), which is an extension of  $L_{\mu}$  with the *past* modalities  $\diamond^{-}\varphi$  and  $\diamond_{\eta}^{-}\varphi$ :

- $(\bar{w}, v, i) \models \diamond^{-}\varphi$  iff  $i > 1$  and  $(\bar{w}, v, i - 1) \models \varphi$ .
- $(\bar{w}, v, i) \models \diamond_{\eta}^{-}\varphi$  iff there is an  $\ell$  such that  $\eta(\ell, i)$  holds and  $(\bar{w}, v, \ell) \models \varphi$ .

Greatest fixed-points  $\nu X.\varphi(X)$  are definable in  $L_{\mu}$  as  $\neg\mu X.\neg\varphi(\neg X)$ . Using greatest fixed-points and  $\Box\varphi$  (defined as  $\neg\diamond\neg\varphi$ ), one can push all negations to atoms in  $L_{\mu}$  formulae. For resulting formulae, an important parameter is the alternation-depth of least and greatest fixed-points [7]. We refer to  $L_{\mu}^k$  as the fragment of  $L_{\mu}$  that consists of formulae of alternation depth at most  $k$  (e.g., the alternation-free fragment is  $L_{\mu}^0$ ).

**Languages and unary queries.** Formulae of  $L_{\mu}$  (without free variables) are satisfied in positions of a nested word, and thus they give rise to classes of *unary queries* that return, for  $\bar{w}$ , the set  $\{i \mid (\bar{w}, i) \models \varphi\}$ . Every  $L_{\mu}$  formula  $\varphi$  without free variables defines a language  $\{\bar{w} \mid (\bar{w}, 1) \models \varphi\}$ . Likewise, every MSO formula  $\varphi(x)$  with one free first-order variable defines a unary query, and every MSO sentence defines a language. In the absence of nesting, it is known [7,20] that a language (of words or  $\omega$ -words) is definable by a  $L_{\mu}$  formula iff it is definable by an MSO sentence (not using relation  $\eta$ ).

### 3 Mu-Calculus over Nested Words

Since NWA generalize finite state automata, the translation from MSO to NWAs is nonelementary. But just as for finite words or trees, one can find equally expressive logical formalisms with better model-checking complexity. We show that the equivalence  $\text{MSO} = L_{\mu}$  extends from words and trees to nested words. It applies not only in sentences evaluated in the first position of a nested word, but more generally to unary queries that select a set of positions. This is relevant for finite nested words viewed as streaming XML documents: while theoretical investigations have mostly looked at the case of sentences [23,8], in practical application one typically needs to evaluate unary queries (e.g. XPath) over such streams [21]. To deal with unary queries, we look at  $L_{\mu}$  with the past, i.e.  $L_{\mu}^{\text{full}}$ , and prove that it is equivalent to MSO unary queries. That is:

**Theorem 1.** *For finite nested words and nested  $\omega$ -words, MSO and  $L_{\mu}^{\text{full}}$  define the same classes of unary queries.*

As a corollary to the proof, we get

**Corollary 1.** *The languages of nested words definable in MSO and  $L_{\mu}$  are the same.*

We can tighten this for finite nested words. Let  $(L_{\mu}^{\text{full}})^+$  be the negation-free (and thus alternation-free) fragment of  $L_{\mu}^{\text{full}}$  that has two additional constants “first” and “last” with their intuitive meanings: “first” holds only at the first position of a nested word, and “last” holds at the last position. Likewise we define  $(L_{\mu})^+$  from  $L_{\mu}$ .

**Corollary 2.** *For unary queries over finite nested words,  $\text{MSO} = L_\mu^{\text{full}} = (L_\mu^{\text{full}})^+$ . Furthermore,  $\text{MSO}$ ,  $L_\mu$ , and  $(L_\mu)^+$  define the same languages of finite nested words.*

From [14], we conclude that for every  $(L_\mu^{\text{full}})^+$  formula  $\varphi$  and every finite nested word  $\bar{w}$ , the set  $\{i \mid (\bar{w}, i) \models \varphi\}$  can be computed in time  $O(|\varphi| \cdot |\bar{w}|)$ .

We make a couple of remarks about the proof of Theorem 1. Nested words are naturally translated into trees, and there is a closely related result in the literature, Niwinski’s theorem, showing that over the full infinite binary tree,  $\text{MSO}$  and  $L_\mu$ , evaluated in the root of the tree, are equally expressive [20]. Despite this, there does not seem to be any easy adaptation of proof techniques in [20] that yields a proof of Theorem 1. Not only do we need a stronger result for unary queries and an extension with the past modalities, but in addition translations of infinite nested words are not complete binary trees.

Another natural attempt at a proof is to use the expressive-completeness result of Janin and Walukiewicz: every bisimulation-invariant  $\text{MSO}$  property is definable in  $L_\mu$  [16]. Then we could express runs of tree automata on tree encodings of nested words by bisimulation-invariant  $\text{MSO}$  sentences, apply [16] to get an equivalent  $L_\mu$  formula for trees, and translate it into an  $L_\mu$  formula over nested words. This sketch indeed can be turned into a proof of  $\text{MSO} = L_\mu$  for languages of nested words, but it breaks already for unary queries over finite nested words, where one needs to encode a more complicated run of a query automaton [19], and it is even harder to adapt this argument to infinite nested words for which we do not have an automaton model capturing unary queries. Thus, our proof is a direct argument based on the *composition method*.

## 4 Automata Models for Nested $\omega$ -Words

**Nested  $\omega$ -word automata.** Visibly pushdown automata (VPA), with both Büchi and Muller acceptance conditions, were introduced in [5], and shown to be equivalent to  $\text{MSO}$ , but not necessarily determinizable. The example of a VPL that cannot be accepted by a deterministic automaton [5] can use arbitrarily many calls without matching returns, something that cannot happen in nested words. Then [18] introduced a notion of *stair visibly pushdown automata* (stair VPA) to control such unmatched calls and showed that stair VPAs are determinizable. These models were defined for VPLs, so we first specialize a particular class of stair VPAs [18] to nested words, thereby obtaining a notion of combined nested word automata, that admit determinization. We then use such automata to show that over nested words, for every  $\omega$ -NWA (with a Büchi or a Muller acceptance condition), there exists an equivalent deterministic Muller  $\omega$ -NWA.

A *combined nested word automaton* (CNWA) puts together an  $\omega$ -word automaton  $\mathcal{A}_1$  with a Muller acceptance condition and a finite NWA  $\mathcal{A}_2$ . It runs  $\mathcal{A}_1$  over all positions that are not inside a call. Every time  $\mathcal{A}_1$  finds a call position  $i$ , it invokes  $\mathcal{A}_2$  to process the finite nested word formed by the elements between  $i$  and its matching return  $j$ , and then it uses its final state to determine what state to assign to  $j + 1$ , and continues its run from position  $j + 1$ . Formally, a CNWA  $\mathcal{A}$  over  $\Sigma$  is a pair  $(\mathcal{A}_1, \mathcal{A}_2)$ , where:

- $\mathcal{A}_2 = (\Sigma, Q_2, Q_2^0, \delta_2 = (\delta_c^2, \delta_r^2, \delta_r^2))$  is an NWA without accepting states;
- $\mathcal{A}_1 = (\Sigma \cup Q_2, Q_1, Q_1^0, \delta_1, \mathcal{F}_1)$  is an  $\omega$ -word automaton over alphabet  $\Sigma \cup Q_2$  (we assume, of course, that  $\Sigma$  and  $Q_2$  are disjoint).

Given a nested  $\omega$ -word  $\bar{w}$  and  $i \geq 1$ , we define the set of *external* positions  $E(\bar{w})$  as positions  $i$  such that there are no  $j, k \geq 1$  such that  $j < i \leq k$  and  $\eta(j, k)$  holds. Note that  $1 \in E(\bar{w})$  and  $E(\bar{w})$  is infinite. If  $i \in E(\bar{w})$  is not a call, then  $i + 1 \in E(\bar{w})$ . If  $i \in E(\bar{w})$  is a call with  $j$  being its matching return, then the next, after  $i$ , element of  $E(\bar{w})$  is  $j + 1$ . With this, we define a *run* of  $\mathcal{A}$  over a nested  $\omega$ -word  $\bar{w} = (a_1 a_2 \dots, \eta)$  as a mapping  $\rho : E(\bar{w}) \rightarrow Q_1$  such that  $\rho(1) \in Q_1^0$  and for every  $i \in E(\bar{w})$ :

- if  $i$  is not a call (and  $i + 1 \in E(\bar{w})$ ), then  $\rho(i + 1) \in \delta_1(\rho(i), a_i)$ ;
- if  $i$  is a call with return  $j$  (and the successor of  $i$  in  $E(\bar{w})$  is  $j + 1$ ), then  $\rho(j + 1) \in \delta_1(\rho(i), q)$ , where  $q$  is a state in  $Q_2$  such that there exists a run  $\rho_2$  of  $\mathcal{A}_2$  over  $\bar{w}[i, j]$  having  $q$  as the last state.

A CNWA  $\mathcal{A}$  accepts  $\bar{w}$  if there is a run  $\rho$  of  $\mathcal{A}$  over  $\bar{w}$  such that  $\text{Inf}(\rho) \in \mathcal{F}_1$ . We say that CNWA  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  is deterministic if both  $\mathcal{A}_1$  and  $\mathcal{A}_2$  are deterministic. Then results in [18] can be restated in this terminology as:

**Proposition 1** ([18]). *Over nested  $\omega$ -words, CNWAs and deterministic CNWAs are equivalent.*

We show, by using standard techniques, that CNWA and MSO are equivalent, from which the main result of this section follows:

**Theorem 2.** *Over nested  $\omega$ -words, MSO,  $\omega$ -NWA and deterministic  $\omega$ -NWA with Muller acceptance condition, define precisely the regular languages. Moreover, translations between these formalisms are effective.*

Determinization of  $\omega$ -NWAs is done by translating them into CNWAs, determinizing those (which involves a  $2^{O(n \log n)}$  Safra construction [22] and a  $2^{O(n^2)}$  determinization procedure from [5]) and then translating back into  $\omega$ -NWAs with Muller acceptance condition. Putting these three components together, we get (note that the bound is the same as for determinization of stair VPAs for VPLs [18]):

**Corollary 3.** *For every  $\omega$ -NWA with  $n$  states, there exists an equivalent deterministic  $\omega$ -NWA with a Muller acceptance condition and with  $2^{O(n^2)}$  states.*

It is well-known that a language of  $\omega$ -words is regular (accepted by a Büchi or a Muller automaton) iff it is a finite union of languages of the form  $UV^\omega$ , where  $U, V$  are regular languages. Automata characterizations imply a similar result for nested  $\omega$ -words.

**Corollary 4.** *A language of nested  $\omega$ -words is regular iff it is a finite union of languages of the form  $UV^\omega$  where  $U$  and  $V$  are regular languages of finite nested words.*

A basic problem in automata theory is the nonemptiness problem: is the language accepted by an automaton nonempty? It was shown in [5], that nonemptiness, and more generally reachability problem for visibly pushdown  $\omega$ -automata, is polynomial. Combining this with a NLOGSPACE algorithm for nonemptiness of  $\omega$ -word automata, we get polynomial nonemptiness algorithms for  $\omega$ -NWA and CNWA. Further, a modification of PTIME-hardness reduction for emptiness for context-free grammars gives us:

**Corollary 5.** *The nonemptiness problem for  $\omega$ -NWA and CNWA is PTIME-complete.*

It is easy to code a deterministic automaton by an  $L_\mu^1$  formula. Thus,

**Corollary 6.** *Over nested  $\omega$ -words,  $L_\mu$  collapses to  $L_\mu^1$ .*

**Alternating automata for nested  $\omega$ -words.** In the context of formal verification, alternating automata have proved to be the key to a comprehensive automata-theoretic framework for temporal logics [27]. With the development of temporal logics for nested words [421], it is natural to develop alternating automata for nested words, with the hope that they can simplify the process of translating temporal logics into automata.

We now define alternating automata for both finite and infinite nested words, and show that they are equivalent to NWA. We note that this is in sharp contrast with the theory of alternating automata for nested trees, where alternating automata are known to be more expressive than nondeterministic automata [3].

First recall the definition of alternating automata for finite and infinite words. Given a set of states  $Q$ , let  $\mathcal{B}^+(Q)$  be the set of positive Boolean combinations of elements from  $Q$ . Given  $X \subseteq Q$  and  $\varphi \in \mathcal{B}^+(Q)$ , we say that  $X$  satisfies  $\varphi$  if the truth assignment  $\sigma_X$  satisfies  $\varphi$ , where  $\sigma_X$  is defined as  $\sigma_X(q) = 1$  iff  $q \in X$ . Then an alternating ( $\omega$ -)word automaton  $\mathcal{A}$  is a tuple  $(\Sigma, Q, Q_0, \delta, F)$ , where  $Q, Q_0$  and  $F$  are defined as for the case of word automata, and  $\delta : Q \times \Sigma \rightarrow \mathcal{B}^+(Q)$  is a transition function.

A run of such an automaton is a labeled tree. A  $\Sigma$ -labeled tree  $T$  is a pair  $(D, \lambda)$ , where  $\lambda : D \rightarrow \Sigma$  and  $D$  is a prefix-closed subset of  $\mathbb{N}^*$  such that (1) if  $x \cdot i \in D$  and  $0 \leq j < i$ , then  $x \cdot j \in D$ , and (2) for every  $x \in D$ , there exists a finite number of strings of the form  $x \cdot i$  in  $D$  (finite branching). For  $x \in \mathbb{N}^*$ , its length is denoted by  $|x|$ . The depth of a tree is  $\max_{x \in D} |x|$ .

A run of an alternating word automaton  $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$  over  $w = a_1 \cdots a_n$  is a finite  $Q$ -labeled tree  $T = (D, \lambda)$  of depth  $n$  such that  $\lambda(\varepsilon) \in Q_0$  and for every  $x \in D$  that has children  $x \cdot 0, \dots, x \cdot \ell$  of length  $i$ , we have that  $\{\lambda(x \cdot 0), \dots, \lambda(x \cdot \ell)\}$  satisfies  $\delta(\lambda(x), a_i)$ . An alternating word automaton  $\mathcal{A}$  accepts a word  $w$  if there is a run  $T = (D, \lambda)$  of  $\mathcal{A}$  over  $w$  such that  $\lambda(x) \in F$  for every node  $x$  in  $T$  of length  $n$ . The run of an alternating  $\omega$ -word automaton  $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$  over an  $\omega$ -word  $w = a_1 a_2 \cdots$  is defined in exactly the same way as an infinite  $Q$ -labeled tree  $T = (D, \lambda)$ . Then  $\mathcal{A}$  accepts  $\omega$ -word  $w$  if there is an accepting run  $T = (D, \lambda)$  of  $\mathcal{A}$  over  $w$ , i.e. every infinite branch  $\rho$  of  $T$  includes infinitely many labels in  $F$ .

An alternating nested word automaton (or alternating NWA, or ANWA) is an NWA that admits alternation in call, return, and internal transitions. Formally, an ANWA  $\mathcal{A}$  is a tuple  $(\Sigma, Q, Q_0, \delta, F)$ , where  $Q, Q_0$  and  $F$  are defined as for the case of alternating word automata, and  $\delta$  is a triple  $(\delta_c, \delta_l, \delta_r)$  of transition functions  $\delta_c, \delta_l : Q \times \Sigma \rightarrow \mathcal{B}^+(Q)$ , and  $\delta_r : Q \times Q \times \Sigma \rightarrow \mathcal{B}^+(Q)$ . A run of  $\mathcal{A}$  over  $\bar{w} = (a_1 \cdots a_n, \eta)$  is a  $Q$ -labeled finite tree  $T = (D, \lambda)$  of depth  $n$  such that  $\lambda(\varepsilon) \in Q_0$  and for every  $x \in D$  with children  $x \cdot 0, \dots, x \cdot \ell$ , of length  $i \leq n$ :

- if  $|x|$  (i.e.  $i - 1$ ) is a call position, then  $\{\lambda(x \cdot 0), \dots, \lambda(x \cdot \ell)\}$  satisfies  $\delta_c(\lambda(x), a_i)$ ;
- if  $|x|$  is an internal position, then  $\{\lambda(x \cdot 0), \dots, \lambda(x \cdot \ell)\}$  satisfies  $\delta_l(\lambda(x), a_i)$ ;
- if  $|x|$  is a return position with matching call  $j$  and  $y$  is the prefix of  $x$  with  $|y| = j - 1$ , then  $\{\lambda(x \cdot 0), \dots, \lambda(x \cdot \ell)\}$  satisfies  $\delta_r(\lambda(x), \lambda(y), a_i)$ .

Note that if  $i - 1$  is an internal position and  $x$  does not have children, then  $\delta_l(\lambda(x), a_i)$  has to be a tautology, and likewise for call and return positions. An alternating nested



word automaton  $\mathcal{A}$  accepts a nested word  $\bar{w}$  if there is a run  $T = (D, \lambda)$  of  $\mathcal{A}$  over  $\bar{w}$  such that  $\lambda(x) \in F$  for every node  $x$  in  $T$  of length  $n$ .

**Proposition 2.** *For every alternating NWA, there exists an equivalent NWA.*

This can be extended to nested  $\omega$ -words. An alternating nested  $\omega$ -word automaton ( $\omega$ -ANWA)  $\mathcal{A}$  is a tuple  $(\Sigma, Q, Q_0, \delta, F)$ , where  $Q, Q_0, \delta$  and  $F$  are defined exactly as for ANWA. A *run* is defined in the same way as above, and the acceptance condition again states that along each infinite branch, states from  $F$  are seen infinitely often.

**Theorem 3.** *For every  $\omega$ -ANWA with  $n$  states, there exists (and can be effectively constructed) an equivalent  $\omega$ -NWA with a Büchi acceptance condition and  $2^{O(n^4)}$  states.*

For the proof, we introduce a notion of *alternating combined nested word automaton* (ACNWA) and provide a direct translation from  $\omega$ -ANWA into ACNWA. Then by using Proposition 2 and the fact that alternating  $\omega$ -word automata can be translated into  $\omega$ -word automata [27], we give a translation from ACNWA into CNWA. Theorem 3 then follows from Proposition 1. We note that Theorem 3 provides an exponential-time algorithm for checking nonemptiness of ANWAs and  $\omega$ -ANWAs. Since even in the finite case the problem is as hard as universality for finite tree automata [24], we get:

**Corollary 7.** *The nonemptiness problem for both ANWAs and  $\omega$ -ANWAs is EXPTIME-complete.*

## 5 Synchronization of Nested Words

Synchronization of words leads to a concept of *regular relations*. It ties together (synchronizes) positions in several words, and then runs an automaton over them [15]. To be concrete, let  $w_1, \dots, w_k$  be words from  $\Sigma^*$ . Assume that  $\#$  is a letter that is not in  $\Sigma$ . Let  $n = \max_i |w_i|$ , and let  $[(w_1, \dots, w_k)]$  be a word of length  $n$  constructed as follows. It is over the alphabet  $(\Sigma \cup \{\#\})^k$ , and its  $i$ th letter is a  $k$ -tuple  $\mathbf{a}_i = (a_1^i, \dots, a_k^i)$ , where each  $a_j^i$  is the  $i$ th letter of  $w_j$  if  $i \leq |w_j|$ , and  $\#$  if  $i > |w_j|$ . That is, we pad words shorter than  $n$  with  $\#$ 's to make them all of length  $n$ , and then take the  $i$ th letter of  $[(w_1, \dots, w_k)]$  to be the tuple of the  $i$ th letters of these padded words.

Then *regular  $k$ -ary relations* over  $\Sigma$  are defined as sets  $R \subseteq (\Sigma^*)^k$  such that the set  $\{[(w_1, \dots, w_k)] \mid (w_1, \dots, w_k) \in R\}$  is accepted by an automaton over the alphabet  $(\Sigma \cup \{\#\})^k$  [13][12]. Such automata are called *letter-to-letter automata*. Regular relations are closed under Boolean combinations, product, and projection. This makes it possible to find infinite structures over  $\Sigma^*$  with decidable first-order theories whose definable sets are precisely the regular relations (these are universal *automatic structures*, cf. [13][12]). The most commonly used such structure is  $\langle \Sigma^*, \prec, (P_a)_{a \in \Sigma}, \text{el} \rangle$ , where  $\prec$  is the prefix relation,  $P_a(w)$  is true iff the last letter of  $w$  is  $a$ , and  $\text{el}(w, w')$  (the equal-length predicate) holds iff  $|w| = |w'|$  [13][12][10].

We now study synchronization for nested words. We show that the usual letter-to-letter synchronization for words is completely incompatible with the nesting structure because even the simplest nested extension of letter-to-letter automata is undecidable.

We propose a different decidable synchronization scheme for nested words that gives us regular relations with all the expected properties.

*Letter-to-letter nested word automata.* Assume that we have  $k$  nested words  $\bar{w}_1, \dots, \bar{w}_k$ , and we again pad the shorter words with a special symbol  $\#$  so that all of them are of the same length  $n$ . Let  $[(\bar{w}_1, \dots, \bar{w}_k)]$  be such a nested word over the alphabet  $(\Sigma \cup \{\#\})^k$ , and let  $\mathbf{a}_i$  be its  $i$ th letter. The letter-to-letter automaton runs from left to right on  $[(\bar{w}_1, \dots, \bar{w}_k)]$ , as an NWA. The main difference with NWAs is that each position  $i$  may now be a return position in *several* of the  $\bar{w}_j$ 's, and thus states in several call positions determine the next state.

That is, in a  $k$ -letter-to-letter NWA over  $k$ -tuples of nested words, we have multiple return transitions  $\delta_r^X : Q \times Q^{|X|} \times (\Sigma \cup \{\#\})^k \rightarrow 2^Q$ , indexed by nonempty  $X \subseteq \{1, \dots, k\}$ . Suppose  $i$  is a return position in  $\bar{w}_{l_1}, \dots, \bar{w}_{l_m}$ , where  $1 \leq l_1 < \dots < l_m \leq k$  and  $m > 0$ . If  $j_1, \dots, j_m$  are the matching calls, i.e.  $\eta_{l_1}(j_1, i), \dots, \eta_{l_m}(j_m, i)$  hold, then  $\rho(i + 1)$  depends on  $\rho(i)$ ,  $\mathbf{a}_i$ , and the states in positions  $j_1, \dots, j_m$ :

$$\rho(i + 1) \in \delta_r^{\{l_1, \dots, l_m\}}(\rho(i), \rho(j_1), \dots, \rho(j_m), \mathbf{a}_i).$$

For positions without returns, we have one transition  $\delta : Q \times (\Sigma \cup \{\#\})^k \rightarrow 2^Q$ .

We show that even a much simpler automaton is undecidable. Define a *simplified  $k$ -letter-to-letter NWA* as a  $k$ -letter-to-letter NWA with one return transition is  $\delta_r : Q \times Q \times (\Sigma \cup \{\#\})^k \rightarrow 2^Q$ , just as in the usual NWA. The condition on the run is as follows: if  $i$  is a return position in words  $\bar{w}_{l_1}, \dots, \bar{w}_{l_m}$ , for  $1 \leq l_1 < \dots < l_m \leq k$ , then  $\rho(i + 1) \in \delta_r(\rho(i), \rho(j_1), \mathbf{a}_i)$ , where  $j_1$  is the call of  $i$  in  $\bar{w}_{l_1}$ . In other words, we look at the state of only one call position, corresponding to the return with the smallest index. For all other positions we have a single transition  $\delta : Q \times (\Sigma \cup \{\#\})^k \rightarrow 2^Q$ .

If  $k = 1$ , these are the usual NWAs. But if  $k = 2$ , they are undecidable.

**Theorem 4.** *The nonemptiness problem is undecidable for simplified 2-letter-to-letter NWAs (and thus for  $k$ -letter-to-letter NWAs for  $k > 1$ ).*

Thus, there is no hope to use even the simplest possible form of letter-to-letter synchronization in nested words. As another example of such incompatibility, we show that there are no natural decidable extensions of universal automatic structures on words to nested words. We look at structures  $\mathfrak{M} = \langle \Sigma_{\text{nw}}^*, \Theta \rangle$  (where  $\Sigma_{\text{nw}}^*$  is the set of all finite nested words over  $\Sigma$ ) of a vocabulary  $\Theta$ . We assume that  $\Theta$  includes some basic relations. One is a prefix relation  $\bar{w} \preceq_{\text{nw}} \bar{w}'$  iff  $\bar{w} = \bar{w}'[1, m]$  for some  $m \leq |\bar{w}'|$  (so we can refer to the linear structure of nested words). The other allows us to refer to the nesting structure: we relate a prefix  $\bar{w}$  of  $\bar{w}'$  so that in  $\bar{w}'$ , there is a call-return edge from the last position of  $\bar{w}$  to the last position of  $\bar{w}'$ . That is,  $\bar{w} \preceq_{\eta} \bar{w}'$  iff  $\bar{w} = \bar{w}'[1, m]$ , and  $\eta(m, |\bar{w}'|)$  holds in  $\bar{w}'$ . We say that  $\mathfrak{M}$  *defines all regular languages of nested words* if for each such language  $L$ , there is a formula  $\varphi_L(x)$  such that  $L = \{\bar{w} \in \Sigma_{\text{nw}}^* \mid \mathfrak{M} \models \varphi(\bar{w})\}$ . We say that  $\mathfrak{M}$  *defines all regular relations over words* if for each regular relation  $R \subseteq (\Sigma^*)^k$ , there is a formula  $\psi_R(x_1, \dots, x_k)$  such that  $\mathfrak{M} \models \psi_R(\bar{w}_1, \dots, \bar{w}_k)$  iff  $(w_1, \dots, w_k) \in R$  (recall that  $w_i$  is a word from  $\Sigma^*$  obtained by removing the nesting structure from  $\bar{w}_i$ ).

**Proposition 3.** *There is no structure  $\mathfrak{M} = \langle \Sigma_{\text{nw}}^*, \preceq_{\text{nw}}, \preceq_{\eta}, \dots \rangle$  that defines all regular languages of nested words, all regular relations over words, and has a decidable theory.*

*Call-return synchronization.* As the usual letter-to-letter synchronization is incompatible with nested words, we propose a different, *call-return* synchronization. Intuitively, instead of synchronizing positions with the same index  $i$  in different words, we synchronize positions for which the shortest paths to them (from the first position) are the same. To formalize this, we use a notion of a *summary path* introduced recently in connection with the study of LTL-like logics on nested  $\omega$ -words [11]. A summary path to a position  $i$  in a nested word  $\bar{w}$  is the shortest path from 1 to  $i$  that combines both successor and matching edges. That is, it is a sequence  $1 = i_0 < i_1 < \dots < i_k = i$  such that, if  $i_l$  is a call with  $\eta(i_l, j)$  and  $i \geq j$ , then  $\eta(i_l, i_{l+1})$  holds, and otherwise  $i_{l+1} = i_l + 1$ . We represent this summary path as a word  $a_1 \dots a_k$  over the alphabet  $A = \{i, c, m\}$ :

1. if  $i_l = i_{l-1} + 1$  and  $i_{l-1}$  is not a call, then  $a_l = i$  (path goes via an internal edge);
2. if  $i_l = i_{l-1} + 1$  and  $i_{l-1}$  is a call, then  $a_l = c$  (path goes via a call edge);
3. if  $\eta(i_{l-1}, i_l)$  holds, then  $a_l = m$  (path goes via a matching edge).

If both  $i_l = i_{l-1} + 1$  and  $\eta(i_{l-1}, i_l)$  hold, we let  $a_l$  be  $m$ . The unique summary path to position  $i$  will be denoted by  $\pi_{\bar{w}}^{\eta}(i) \in A^*$ , and the set of all summary paths by  $II^{\eta}(\bar{w})$ . The label of  $\pi_{\bar{w}}^{\eta}(i)$  is the label of  $i$  in  $\bar{w}$ . Note that  $II^{\eta}(\bar{w})$  is closed under prefix.

The idea of the *call-return synchronization* is that now with each position  $i$ , we keep its summary paths  $\pi_{\bar{w}}^{\eta}(i)$ , to remember how it was reached in different nested words. That is, a call-return synchronization of nested words  $\bar{w}_1, \dots, \bar{w}_k$  is a pair  $(II^{\eta}(\bar{w}_1, \dots, \bar{w}_k), \lambda)$  where  $II^{\eta}(\bar{w}_1, \dots, \bar{w}_k) = \bigcup_l II^{\eta}(\bar{w}_l)$ , and  $\lambda : II^{\eta}(\bar{w}_1, \dots, \bar{w}_k) \rightarrow (\Sigma \cup \{\#\})^k$  is a labeling function that labels each summary path with its label in  $\bar{w}_i$  if it occurs in  $\bar{w}_i$ , and with  $\#$  otherwise, for each  $i \leq k$ . This synchronization can naturally be viewed as a tree.

As an example, consider two nested words below,  $\bar{w}_1$  (on the left) and  $\bar{w}_2$  (on the right), with summary paths shown above positions.



The synchronization occurs in the first and the second position, and we recursively synchronize the calls (from  $i$ ) and what follows their returns (from  $im$ ). Intuitively, this results in adding a dummy internal node  $ici$  inside the call for  $\bar{w}_2$ , and adding a dummy last internal position  $imii$  for  $\bar{w}_2$ . Note that position 4 (i.e.  $ici$ ) in  $\bar{w}_1$  is in no way related to position 4 ( $im$ ) in  $\bar{w}_2$ , as it would have been in letter-to-letter synchronization.

We now say that  $R \subseteq (\Sigma_{nw}^*)^k$  is a *regular  $k$ -ary relation of nested words* iff there is a tree automaton on ternary trees that accepts precisely  $(II^{\eta}(\bar{w}_1, \dots, \bar{w}_k), \lambda)$ , for  $(\bar{w}_1, \dots, \bar{w}_k) \in R$ . The following is an immediate consequence of coding tree representations in MSO, and of the work on automatic structures over trees [11]:

- Proposition 4.** – *Regular relations of nested words are closed under union, intersection, complementation, product, and projection.*
- *Regular 1-ary relations of nested words are precisely the regular nested languages.*
  - *There is a finite collection  $\Theta$  of unary and binary predicates on  $\Sigma_{nw}^*$  such that  $\langle \Sigma_{nw}^*, \Theta \rangle$  is a universal automatic structure for nested words, i.e. its definable relations are precisely the regular relations of nested words, and its theory is decidable.*

*Acknowledgments.* We thank Rajeev Alur, Kousha Etessami, and Neil Immerman for helpful discussions. Arenas was supported by FONDECYT grants 1050701, 7060172 and 1070732; Arenas and Barceló by grant P04-067-F from the Millennium Nucleus Centre for Web Research; Libkin by the EC grant MEXC-CT-2005-024502, EPSRC grant E005039, and by an NSERC grant while on leave from U. Toronto.

## References

1. Alur, R., Arenas, M., Barceló, P., Etessami, K., Immerman, N., Libkin, L.: First-order and temporal logics for nested words. In: LICS 2007
2. Alur, R., Chaudhuri, S., Madhusudan, P.: A fixpoint calculus for local and global program flows. In: POPL 2006, pp. 153–165.
3. Alur, R., Chaudhuri, S., Madhusudan, P.: Languages of nested trees. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 329–342. Springer, Heidelberg (2006)
4. Alur, R., Etessami, K., Madhusudan, P.: A temporal logic of nested calls and returns. In: Jensen, K., Podolski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 467–481. Springer, Heidelberg (2004)
5. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: STOC 2004, pp. 202–211.
6. Alur, R., Madhusudan, P.: Adding nesting structure to words. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 1–13. Springer, Heidelberg (2006)
7. Arnold, A., Niwinski, D.: Rudiments of  $\mu$ -calculus. North-Holland, Amsterdam (2001)
8. Bárány, V., Löding, C., Serre, O.: Regularity problems for visibly pushdown languages. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 420–431. Springer, Heidelberg (2006)
9. Barceló, P., Libkin, L.: Temporal logics over unranked trees. In: LICS 2005, pp. 31–40.
10. Benedikt, M., Libkin, L., Schwentick, T., Segoufin, L.: Definable relations and first-order query languages over strings. J. ACM 50(5), 694–751 (2003)
11. Benedikt, M., Libkin, L., Neven, F.: Logical definability and query languages over ranked and unranked trees. In: ACM TOCL. Extended abstract in LICS'02 and LICS'03, vol. 8(2), ACM Press, New York (2007)
12. Blumensath, A., Grädel, E.: Automatic structures. In: LICS 2000, pp. 51–62.
13. Bruyère, V., Hansel, G., Michaux, C., Villemaire, R.: Logic and  $p$ -recognizable sets of integers. Bull. Belg. Math. Soc. 1, 191–238 (1994)
14. Cleaveland, R., Steffen, B.: A linear-time model-checking algorithm for the alternation-free modal  $\mu$ -calculus. In: Larsen, K.G., Skou, A. (eds.) CAV 1991. LNCS, vol. 575, pp. 48–58. Springer, Heidelberg (1992)
15. Elgot, C., Mezei, J.: On relations defined by generalized finite automata. IBM J. Res. Develop. 9, 47–68 (1965)
16. Janin, D., Walukiewicz, I.: On the expressive completeness of the propositional  $\mu$ -calculus with respect to monadic second order logic. In: Sassone, V., Montanari, U. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 263–277. Springer, Heidelberg (1996)
17. Lautemann, C., Schwentick, T., Thérien, D.: Logics for context-free languages. In: Pacholski, L., Tiuryn, J. (eds.) CSL 1994. LNCS, vol. 933, pp. 205–216. Springer, Heidelberg (1995)
18. Löding, C., Madhusudan, P., Serre, O.: Visibly pushdown games. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 408–420. Springer, Heidelberg (2004)
19. Neven, F., Schwentick, Th: Query automata over finite trees. TCS 275, 633–674 (2002)
20. Niwinski, D.: Fixed points vs. infinite generation. In: LICS 1988, pp. 402–409

21. Peng, F., Chawathe, S.: Xpath queries on streaming data. In: SIGMOD 2003, pp. 431–442.
22. Safra, S.: On the complexity of omega-automata. In: FOCS 1988, pp. 319–327
23. Segoufin, L., Vianu, V.: Validating streaming XML documents. In: PODS 2002, pp. 53–64.
24. Seidl, H.: Deciding equivalence of finite tree automata. SICOMP 19(3), 424–437 (1990)
25. Thomas, W.: Languages, automata, and logic. Handbook of Formal Languages, vol. 3 (1997)
26. Thomas, W.: Infinite trees and automaton-definable relations over  $\omega$ -words. TCS 103, 143–159 (1992)
27. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. Banff Higher Order Workshop, pp. 238–266 (1995)
28. Vardi, M.Y.: Reasoning about the past with two-way automata. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 628–641. Springer, Heidelberg (1998)

# A Combinatorial Theorem for Trees

## Applications to Monadic Logic and Infinite Structures

Thomas Colcombet

Cnrs/Irisa

thomas.colcombet@irisa.fr

**Abstract.** Following the idea developed by I. Simon in his theorem of Ramseyan factorisation forests, we develop a result of ‘deterministic factorisations’. This extra determinism property makes it usable on trees (finite or infinite).

We apply our result for proving that, *over trees*, every monadic interpretation is equivalent to the composition of a first-order interpretation (with access to the ancestor relation) and a monadic marking. Using this remark, we give new characterisations for prefix-recognisable structures and for the Caucal hierarchy.

Furthermore, we believe that this approach has other potential applications.

**Topics:** Semigroups, Ramseyan factorisation, Monadic second-order logic, Trees, Infinite structures.

## 1 Introduction

The theorem of factorisation forests was proposed by Simon [20]. One way to present it is the following. For every semigroup morphism  $\varphi$  from  $A^+$  to some finite semigroup  $S$ , there exists a regular expression evaluating to  $A^+$  in which the Kleene exponent  $L^*$  is allowed only when  $\varphi(L) = \{e\}$  for some  $e = e^2 \in S$ ; i.e., the Kleene star is allowed only if it produces a Ramseyan factorisation of the word. The original statement is slightly different in its presentation. It establishes the existence of a so called ‘Ramseyan factorisation of bounded depth’ for every word; those factorisations intuitively witness the acceptance of the word by the regular expression mentioned above. The present paper is based on the proof of the theorem of factorisation forests in [9] which is a simplification of the original presentation.

The result itself has been used for various applications. In [21], Simon uses this theorem for studying the finiteness problem of regular languages of matrices over the tropical semiring (i.e. the semiring  $\mathbb{N} \cup \{\infty\}$  equipped with the minimum and addition operations). This problem is equivalent to the limitedness problem for distance automata. This question is at the heart of the very difficult proof of decidability of the star-height problem due to Hashigushi [11] (the star-height problem consists in determining how many nesting of Kleene stars are required for describing a given regular language of words by a regular

expression). In [16] the theorem is used in a characterisation of the polynomial closure of a variety of languages. In [2], the authors use the theorem of factorisation forests in a complementation result extending the one of Büchi over infinite words. A direct consequence of the result in [2] is the decidability of the limit-edeness problem for nested distance desert automata: this problem extends the one for distance automata seen above, and is the cornerstone of the modern and much simpler solution to the star-height problem proposed by Kirsten [12]. In general the theorem of factorisation forests entails very deep consequences in the understanding of the structure of semigroups. For instance, one directly derives from it a constructive proof of Brown’s lemma on locally finite semigroups [3].

Independently of the contributions of this paper, let us advertise the importance of the factorisation forests theorem. This result is well known in semigroup theory, in which some of its consequences are investigated. But this theorem clearly has other potential fields of application. In the present paper for instance, we use the approach for an application in logic. The interest of this theorem is in fact more natural outside the scope of semigroup theory: for a non-specialist in semigroups, it happens to be much easier to use than to prove. Thus, it is hardly avoidable in some situations (such as in [2]).

The present paper is an attempt to adapt the theorem of factorisation forests in a framework suitable for its use on trees. Essentially the problem we have concerning the original statement is the following: given two words sharing a common prefix, the factorisation forests theorem explicits the existence of a factorisation for each of the two words, but those two factorisations need not coincide on the common prefix. For eliminating this problem, we introduce an extra determinism requirement: the original theorem shows the existence of a factorisation for every word; our theorem shows the existence of a factorisation which is computable ‘deterministically, on-line’ while reading the word from left to right. For reaching this goal, we modify in two ways the original result. A cosmetic modification is that we drop the original formalism using trees — determinism does not fit naturally in it —, and replace it by the notion of splits for representing factorisation (see below). The second modification consists in weakening the hypothesis of ‘Ramseyanity’, and replace it by a notion of ‘forward Ramseyanity’. Without this weakening, the result would simply not hold.

The second part of the paper is devoted to an application of this result to monadic (second-order) logic over trees. This result has been chosen as an application because it is new, because it does not contain too much technicalities, and also because it could not be derived from weaker version of the main theorem. Let us recall that the monadic logic is an extension of first-order logic by the possibility to quantify over sets of elements. Over words, trees as well as infinite words and trees (of length/height  $\omega$ ), the expressivity of closed monadic formulæ coincide with the standard classes of automata ([5,4,17]). In particular, this logic is known to be more expressive than first-order logic, already over words. We use our result to decompose monadic formulæ over trees: every monadic formula with only free first-order variables is equivalent *over trees* to a first-order formula with access to the ancestor relation and to monadically defined unary

predicates. Equivalently, every monadic interpretation is equivalent, over trees, to the composition of a first-order interpretation and a monadic marking.

We apply this result to the theory of infinite structures. We give new characterisations to the class of prefix-recognisable structures as well as to the Caucal hierarchy.

## 2 Main Result

We first define semigroups and additive labellings, then words in Sections 2.1 and 2.2. Our main result is presented in Section 2.3.

### 2.1 Semigroups and Additive Labellings

A *semigroup*  $(S, \cdot)$  is a set  $S$  equipped with an associative binary operator written multiplicatively. Groups and monoids are particular instances of semigroups. A *morphism of semigroup* from a semigroup  $(S, \cdot)$  to a semigroup  $(S', \cdot')$  is a mapping  $\varphi$  from  $S$  to  $S'$  such that for all  $x, y$  in  $S$ ,  $\varphi(x \cdot y) = \varphi(x) \cdot' \varphi(y)$ . An *idempotent* in a semigroup is an element  $e$  such that  $e^2 = e$ .

Let us recall that a *linear ordering*  $(\alpha, <)$  is a set  $\alpha$  together with a total strict ordering relation  $<$ . Let  $\alpha$  be a linear ordering and  $(S, \cdot)$  be a semigroup. A mapping  $\sigma$  from couples  $(x, y)$  with  $x, y \in \alpha$  and  $x < y$  to  $S$  is called an *additive labelling* if for every  $x < y < z$  in  $\alpha$ ,  $\sigma(x, y) \cdot \sigma(y, z) = \sigma(x, z)$ .

### 2.2 Words

Given an alphabet  $A$ , we denote by  $A^*$  the set of finite words over  $A$ , i.e. finite sequences of letters in  $A$ . The length of the word is the length of the sequence. The empty word is  $\varepsilon$ , and  $A^+$  represents  $A^* \setminus \{\varepsilon\}$ .  $A^+$  equipped with the concatenation of words is a semigroup. Given a word  $u$  of length  $n$ , and  $i, j$  with  $0 \leq i \leq j \leq n$ ,  $u_{i,j}$  is the word  $u_{i+1}u_{i+2} \dots u_j$ . Given a finite semigroup  $S$ , a morphism of semigroup  $\varphi$  from  $A^+$  to  $S$ , and a word  $u$  of length  $n$ ,  $\varphi_u$  is the *additive labelling* from  $[0, n]$  to  $S$  defined by

$$\varphi_u(i, j) = \varphi(u_{i,j}).$$

Reciprocally, given an additive labelling  $\sigma$  over  $([0, n], <)$  for some  $n$ , one can associate the word  $\langle \sigma \rangle$  of length  $n$  over the alphabet  $S$ , the  $i$ th letter of which is  $\sigma(i - 1, i)$ . Of course,  $\varphi_{\langle \sigma \rangle} = \sigma$ , where  $\varphi$  is the canonical semigroup morphism from  $S^+$  to  $S$ . According to this remark, additive labellings and words together with a semigroup morphism form two sides of the same object.

### 2.3 Main Theorem

A *split of height  $N$*  of a linear ordering  $\alpha$  is a mapping  $s$  from  $\alpha$  to  $[1, N]$  (we use square brackets for intervals of natural numbers). Given a split, two elements  $x$  and  $y$  in  $\alpha$  such that  $s(x) = s(y) = k$  are  *$k$ -neighbours* if  $s(z) \geq k$  for all  $z \in [x, y]$ .



$k$ -neighbourhood is an equivalence relation over  $s^{-1}(k)$ . A split  $s$  of height  $N$  is *forward Ramseyan* wrt.  $\sigma$  if for every  $k = 1 \dots n$  and every  $x, y, x', y'$  in the same class of  $k$ -neighbourhood with  $x < y$  and  $x' < y'$ ,

$$\sigma(x, y) = \sigma(x, y) \cdot \sigma(x', y') . \tag{1}$$

So in particular,  $\sigma(x, y)$  is an idempotent, but  $\sigma(x, y)$  and  $\sigma(x', y')$  may be different idempotents<sup>1</sup>. Our main result is the following.

**Theorem 1.** *Fix a finite semigroup  $(S, \cdot)$ , an alphabet  $A$  and a semigroup morphism  $\varphi$  from  $A^+$  to  $S$ . There is a partition of  $A^*$  into regular languages  $L_1, \dots, L_K$  with  $K \leq |S|$  such that for every word  $u$  of length  $n$ ,  $s_u$  defined by*

$$\text{for all } i \in [0, n], \quad s_u(i) = k \quad \text{such that } u_{0,i} \in L_k,$$

*is forward Ramseyan for  $\varphi_u$ .*

The proof essentially follows the one of Chalopin and Leung [9]. In particular, it is based on a decomposition of the semigroup following Green’s relations. Green’s relation reflects the interplay of ideals in a semigroup [10], and their use provides deep informations on the structure of the semigroup.

*Example 1.* For simplicity, we identify  $A$  with  $S$ , and  $\varphi$  is the identity over letters. Let  $S$  be  $\{a, b, c\}$  together with the product defined by  $a = ab = aa$ ,  $b = ba = bb$  and  $c = cc = ac = bc = ca = cb$ , then the languages

$$L_1 = \varepsilon + (a + b)^*c, \quad \text{and} \quad L_2 = (a + b + c)^*(a + b)$$

form a valid output of the theorem. For instance, consider the word  $abcbaabacbaa$ , the split defined is (the value of the split being interleaved in the word):

$$1 a 2 b 2 c 1 b 2 a 2 a 2 b 2 a 2 c 1 b 2 a 2 a 2 .$$

Given two 1-neighbour positions  $i < j$ , the letter just before  $j$  is  $c$ ; this means  $\varphi_u(i, j) = c$ . While given two 2-neighbour positions  $i < j$ , all letters in  $u_{i,j}$  belong to  $\{a, b\}$ ; this means  $\varphi_u(i, j) \in \{a, b\}$ . Those remarks entail the forward Ramseyanity since  $c$  is an idempotent, and  $a, b$  are idempotents satisfying  $ab = a$  and  $ba = b$ .

### 3 Application to Monadic Second-Order Logic

We first define structures, graphs and trees in Section 3.1. Then Section 3.2 introduces logics. Section 3.3 presents our result, Theorem 2, and Section 3.4 studies its consequences over infinite structures.

---

<sup>1</sup> In terms of Green’s relation,  $\sigma(x, y)$  and  $\sigma(x', y')$  are  $\mathcal{L}$ -equivalent idempotents.

### 3.1 Structures

**Structures.** A (*relational*) structure  $(\mathcal{U}, R_1, \dots, R_n)$  is a set  $\mathcal{U}$ , called the *universe*, together with *relations*  $R_1, \dots, R_n$  of fixed finite arity over  $\mathcal{U}$ . Each relation  $R$  has a *name* that we write  $R$  itself. The relation is called the *interpretation* of the name in the structure. The *signature* of a structure contains the names involved together with their arity. By extension, we allow (partial) functions from  $\mathcal{U}^k$  to some finite set  $E = \{e_1, \dots, e_n\}$ . Such a function  $f$  is nothing but a shorthand for using  $n$   $k$ -ary relations  $F_1, \dots, F_n$ , each  $F_i$  being interpreted as  $f^{-1}(e_i)$ .

**Words.** Our base structure is  $([0, n], <)$ , i.e. the natural numbers equipped with the natural ordering. An additive labelling  $\sigma$  on it to some finite semigroup can be directly represented in it according to the remark above: this provides the structure  $([0, n], <, \sigma)$ . Our coding of a word  $u$  of length  $n$  is slightly non-standard. It is the structure  $([0, n], <, u)$  where  $u$  is a partial mapping from  $[1, n]$  to  $A$ . The element 0 of this structure is unused. This makes it easier to jump from  $([0, n], <, \sigma)$  to  $([0, n], <, \langle \sigma \rangle)$  and vice versa.

**Graphs.** A (*directed*) *graph* is a structure for which all relations have arity 1 but one of arity 2 called the *edge relation*. The elements of the universe are called *vertices*, the unary relations are *labelling relations*. A *path* is a finite sequence of vertices such that two successive vertices are in relation by the edge relation. The first vertex is called the *origin* of the path, and the last vertex the *destination*.

**Trees.** A *tree*  $t$  is a graph for which the edge relation is called the *ancestor relation*, is denoted by  $\sqsubseteq$ , and satisfies:

- the relation  $\sqsubseteq$  is an order,
- there is a minimal element for  $\sqsubseteq$ , called the *root*,
- for every  $u$ , the set  $\{v : v \sqsubseteq u\}$  is finite and totally ordered.

The vertices of a tree are called *nodes*. Maximal chains are called *branches*. The maximal element smaller than node  $u$  is called (when it exists) the *parent* of  $u$ .

We extend the notions for words to trees: an *additive labelling* is a mapping  $\sigma$  from pairs of nodes  $(x, y)$  such that  $x \sqsubset y$  to a finite semigroup  $S$  which is an additive labelling when restricted to every branch. We also note by  $\langle \sigma \rangle$  the partial function from nodes different from the root to  $S$  defined by  $\langle \sigma \rangle(u) = \sigma(v, u)$  for  $v$  the parent of  $u$  (if it exists). A *split* of height  $N$  is a mapping from nodes to  $[1, N]$ . The split is *forward Ramseyan* wrt.  $\sigma$  if it is forward Ramseyan wrt.  $\sigma$  over every branch.

*Caution:* The trees are *not* defined by a ‘direct successor’ relation, but rather by the ancestor relation. This has major impact on the logic side: all the logics we use below can refer to the ancestor relation, and it is well-known that first-order logic using this ancestor relation is significantly more expressive over trees than first-order logic with access to the successor of a node only. The ancestor relation is necessary in this work.

The *complete binary tree* has as universe  $\{0, 1\}^*$ , as ancestor relation the prefix relation, and has two unary relations,  $0 = \{0, 1\}^*0$  and  $1 = \{0, 1\}^*1$ . We call the relation 0 the *left-child relation*, while 1 is the *right-child relation*. We denote by  $\Delta_2$  the complete binary tree.

One constructs a tree from a graph by unfolding. Given a graph  $G$  and one of its vertices  $v$ , the *unfolding* of  $G$  from  $v$  is the tree which has as nodes the paths of origin  $v$ , as ancestor relation the prefix relation over paths, and such that a path  $\pi$  is labelled by  $a$  in the unfolding iff its destination is labelled by  $a$  in the graph.

### 3.2 Logics

*First-order logic.* We assume a countable set of *first-order variables*  $x, y, \dots$  to pick from. The *atomic formulæ* are  $R(x_1, \dots, x_n)$  for  $x_1, \dots, x_n$  first-order variables and  $R$  a name of relation of arity  $n$ ; given two first-order variables  $x, y$ ,  $x = y$  is also an atomic formula. *First-order logic formulæ* are made out of atomic formulæ combined by the boolean connectives  $\vee, \wedge, \neg$ , and the first-order quantifiers  $\exists x$  and  $\forall x$ .

*Monadic logic.* We assume a countable set of *monadic variables*  $X, Y, \dots$ . *Monadic (second-order) formulæ* are defined as first-order formulæ, but further allow the use of monadic quantifiers  $\exists X, \forall X$ , and of a membership atomic formula  $x \in X$ , where  $x$  is first-order and  $X$  monadic.

We use the standard notion of *free variables*. A formula without free variables is *closed*. We denote by  $\mathcal{S} \models \phi$  the fact, for a closed formula  $\phi$  and a structure  $\mathcal{S}$ , that the formula is true over the structure  $\mathcal{S}$ . The value of first-order variables range over elements of the universe of the structure, while monadic variables take as values subsets of the universe. For  $\mathcal{S} \models \phi$ , we also say that  $\mathcal{S}$  is a *model* of  $\phi$ , or that  $\phi$  is *satisfied* over  $\mathcal{S}$ . We also allow ourselves to write  $\phi(x_1, \dots, x_n)$  to denote that the free-variables of  $\phi$  are among  $\{x_1, \dots, x_n\}$ . Then given elements  $u_1, \dots, u_n$  in the universe of a structure  $\mathcal{S}$ , we write  $\mathcal{S} \models \phi(u_1, \dots, u_n)$  if the formula  $\phi$  is true over the structure  $\mathcal{S}$ , using the valuation mapping  $x_i$  to  $u_i$ .

A structure  $\mathcal{S}$  has a *decidable L-theory* (where  $L$  is either first-order or monadic), if there is an algorithm which, given a formula  $\phi$  of the logic  $L$ , answers whether  $\mathcal{S} \models \phi$  or not.

**Definability.** Let  $R$  be a relation of arity  $k$  over the universe of some structure  $\mathcal{S}$ . It is said *L definable* (where  $L$  is either first-order or monadic) if there exists an  $L$  formula  $\phi(x_1, \dots, x_k)$  such that  $R(u_1, \dots, u_k)$  iff  $\mathcal{S} \models \phi(u_1, \dots, u_k)$ . *Implicitly*, when we refer to definability, we mean that the formula does not depend of the structure.

A structure  $\mathcal{S}'$  is *L definable* in  $\mathcal{S}$  if its universe is an  $L$  definable subset of the universe of  $\mathcal{S}$ , and all the relations in  $\mathcal{S}'$  are  $L$  definable in  $\mathcal{S}$ . We write  $\mathcal{S}' \leq_{FO} \mathcal{S}$  (*resp.*  $\mathcal{S}' \leq_{MSO} \mathcal{S}$ ) if  $\mathcal{S}'$  is first-order (*resp.* monadically) definable in  $\mathcal{S}$ . The special case  $\mathcal{S}' \leq^1_{MSO} \mathcal{S}$  signifies that  $\mathcal{S}'$  is the structure  $\mathcal{S}$  augmented with some new monadically definable unary relations. We also say that  $\mathcal{S}'$  is

obtained by a *monadic marking* from  $\mathcal{S}$ . The relations  $\lesssim_{FO}$ ,  $\lesssim_{MSO}$  and  $\lesssim_{MSO}^1$  correspond to  $\leq_{FO}$ ,  $\leq_{MSO}$  and  $\leq_{MSO}^1$  respectively, up to isomorphism.

### 3.3 Main Result

We prove in this section Theorem 2 every monadic interpretation is the composition of a first-order interpretation with a monadic marking. For simplicity, we use the notations  $\leq_{FO}$ ,  $\leq_{MSO}$ , etc. But let us stress that the constructions are uniform in the sense that implicitly the formulæ involved do not depend on the structures, but only on their signatures.

We need two intermediate lemmas. Both can be proved using elementary compositional methods (see e.g., [19]). The first one establishes that a monadic formula with many first-order variables can be “first-order” reconstructed out of monadic formulæ of two free first-order variables.

**Lemma 1.** *Every monadic formula  $\Phi(x_1, \dots, x_n)$  is equivalent over trees to a first-order formula with access to binary relations defined by monadic formulæ of the form  $x \sqsubset y \wedge \Psi(x, y)$ .*

The second lemma states that a monadic formula of two free first-order variables can be seen as a monadically definable additive labelling.

**Lemma 2.** *For every monadic formula of the form  $\Phi(x, y)$  of free first-order variables  $x, y$ , there exists a finite semigroup  $S_\Phi$ , a subset  $A_\Phi \subseteq S_\Phi$ , and an additive labelling  $\sigma$  monadically definable in every tree  $t$  such that*

$$\text{for every nodes } u \sqsubset v \text{ in } t, \quad t \models \Phi(u, v) \text{ iff } \sigma(u, v) \in A_\Phi .$$

The combination of the two previous lemmas yields the following.

**Corollary 1.** *For every relation  $R$  monadically definable in some tree  $t$ ,*

$$(t, R) \leq_{FO} (t, \sigma) \leq_{MSO} t ,$$

where  $\sigma$  is an additive labelling from  $t$  to some finite semigroup.

*Proof.* (Idea) If  $R$  is defined by a monadic formula of the form  $x \sqsubset y \wedge \Psi(x, y)$ , this is a direct application of Lemma 2. Else, we decompose the formula defining  $R$  as in Lemma 1, and use the argument above for coding each formula of the form  $x \sqsubset y \wedge \Psi(x, y)$ . Each time we obtain a semigroup and an additive labelling. By product, we combine all those informations into a single semigroup and a single additive labelling  $\sigma$ . □

The following lemma is the key argument. It shows how to first-order reconstruct an additive labelling  $\sigma$  out of its unary presentation  $\langle \sigma \rangle$ , providing a forward Ramseyan split is given.

**Lemma 3.** *Fix a semigroup  $S$ . Then*

$$([0, n], <, \sigma) \leq_{FO} ([0, n], <, \langle \sigma \rangle, s)$$

where  $n$  is some natural number,  $\sigma$  an additive labelling from  $[0, n]$  to  $S$ , and  $s$  a split of  $[0, n]$  of height at most  $|S|$  forward Ramseyan wrt.  $\sigma$ .

*Proof.* We have to first-order define  $\sigma$  in  $([0, n], <, \langle \sigma \rangle, s)$ . By a downward induction on  $k = |S| + 1, \dots, 1$ , we construct a function  $a_k(i, j)$  to  $S$  such that for every  $i < j$  in  $[0, n]$ ,  $a_k(i, j) = \sigma(i, j)$  whenever  $s(x) \geq k$  for all  $x \in ]i, j[$ .

For  $k = |S| + 1$ , then  $j = i + 1$ . And  $a_{|S|+1}(i, j) = \langle \sigma \rangle(j) = \sigma(i, j)$ .

For  $k \leq |S|$ . Let  $i < j$  lie in  $[0, n]$ , define

$$a_k(i, j) = \begin{cases} a_{k+1}(i, j) & \text{if } s(x) > k \text{ for all } x \in ]i, j[ , \\ a_{k+1}(i, x).a_{k+1}(x, j) & \text{if } x \text{ is the only element in } ]i, j[ \\ & \text{such that } s(x) = k, \\ a_{k+1}(i, x).a_{k+1}(x, y).a_{k+1}(z, j) & \text{for } x < y \leq z \text{ in } ]i, j[ \\ & \text{with } s(x) = s(y) = s(z) = k \\ & \text{and } s(w) > k \text{ for all } w \in ]i, x[\cup]x, y[\cup]y, z[, j[ . \end{cases}$$

Let  $i < j$  be in  $[0, n]$  such that  $s(x) \geq k$  for all  $x \in ]i, j[$ . We have to show  $a_k(i, j) = \sigma(i, j)$ . In the two first cases, the correctness is obvious. In the last case,  $x, y, z$  are  $k$ -neighbours. Hence, by forward Ramseyanity of  $s$ ,  $\sigma(x, z) = \sigma(x, y).\sigma(y, z) = \sigma(x, z)$  (this holds even if  $y = z$ ). Hence

$$\begin{aligned} a_k(i, j) &= a_{k+1}(i, x).a_{k+1}(x, y).a_{k+1}(z, j) \\ &= \sigma(i, x).\sigma(x, y).\sigma(z, j) \\ &= \sigma(i, x).\sigma(x, z).\sigma(z, j) \\ &= \sigma(i, j) . \end{aligned}$$

Those constructions are first-order definable. The result follows. □

The determinism allows to transfer easily this result to trees.

**Corollary 2.** *Fix a semigroup  $S$ . Then*

$$(t, \sigma) \leq_{FO} (t, \langle \sigma \rangle, s)$$

for a tree  $t$ , an additive labelling  $\sigma$  from  $t$  to  $s$ , and a split of  $t$  of height  $|S|$  forward Ramseyan for  $\sigma$ .

*Proof.* (Idea) The formula defining  $\sigma(u, v)$  for  $u \sqsubset v$  is obtained by relativisation of the formula obtained by Lemma 3 to  $\{w : w \sqsubseteq v\}$ . □

**Lemma 4.** *Let  $R$  be a relation monadically definable in a tree  $t$ ,*

$$(t, R) \leq_{FO} t' \leq_{MSO}^1 t \text{ for some tree } t'.$$

*Proof.* From Corollary 1 it is sufficient to derive from  $(t, \sigma) \leq_{MSO} t$  that

$$(t, \sigma) \leq_{FO} t' \leq_{MSO}^1 t$$

where  $\sigma$  is an additive labelling to a finite semigroup  $S$ . Let  $L_1, \dots, L_{|S|}$  be as in Theorem 1. Set  $s$  to be the split of  $t$  defined by  $s(u) = n$  such that  $t|_{\{v : v \sqsubseteq u\}} \in L_n$

(we see here  $t|_{\{v : v \sqsubseteq u\}}$  as a word, up to isomorphism). By Theorem 1,  $s$  is forward Ramseyan wrt.  $\sigma$ . Furthermore, from the equivalence between regular languages and monadic logic,  $s$  is monadically definable in  $(t, \sigma)$ . Obviously,  $\langle \sigma \rangle$  is also monadically definable in  $(t, \sigma)$ . Combined with Corollary 2 we obtain:

$$(t, \sigma) \leq_{FO} (t, \langle \sigma \rangle, s) \leq_{MSO} (t, \sigma) \leq_{MSO} t .$$

Hence,

$$(t, \sigma) \leq_{FO} (t, \langle \sigma \rangle, s) \leq_{MSO}^1 t . \quad \square$$

By extension of Lemma 4 to structures, we obtain the expected theorem.

**Theorem 2.** *If  $\mathcal{S} \leq_{MSO} t$  then  $\mathcal{S} \leq_{FO} t' \leq_{MSO}^1 t$  for some tree  $t'$ .*

### 3.4 Consequences for Infinite Structures

The goal of this section is to show how Theorem 2 has direct new consequences in the definition of some families of finitely presentable infinite structures: Theorems 4 and 5. But we do not intend to survey this area.

Prefix-recognisable graphs were introduced in 7. Fix a finite alphabet  $A$ . A *prefix-recognisable* graph is an (possibly infinite) graph defined as follows. Its set of vertices is a regular language over the alphabet  $A$ . And each edge relation is a finite union of relations of the form  $(U \times V).W$  with

$$(U \times V).W = \{(uw, vw) : u \in U, v \in V, w \in W\} ,$$

for  $U, V, W$  regular languages. By extension, a graph is *prefix-recognisable* if it is isomorphic to such a graph. An important property of those graphs is that their monadic theory is decidable (this fact is due to Caucal 7; it can be easily seen as a direct consequence of Rabin Theorem 17 stating that the complete binary tree has a decidable monadic theory, together with Theorem 3 below).

There exists different characterisations for this class of graphs. We will use below the following one due to Blumensath 11:

**Theorem 3.** *A graph  $G$  is prefix-recognisable iff  $G \lesssim_{MSO} \Delta_2$ .*

Following this idea, one extends prefix-recognisability to structures: a structure  $\mathcal{S}$  is *prefix-recognisable* if  $\mathcal{S} \lesssim_{MSO} \Delta_2$ .

Theorem 4 provides another – new – characterisation to the prefix-recognisable structures. Beforehand, we need Lemma 5 stating that every regular binary tree is first-order definable in  $\Delta_2$ .

**Lemma 5.** *If  $t \leq_{MSO}^1 \Delta_2$  then  $t \lesssim_{FO} \Delta_2$ .*

And we obtain.

**Theorem 4.** *A structure  $\mathcal{S}$  is prefix-recognisable iff  $\mathcal{S} \lesssim_{FO} \Delta_2$ .*

*Proof.* From  $\mathcal{S} \lesssim_{MSO} \Delta_2$  and Theorem 2,  $\mathcal{S} \lesssim_{FO} t' \leq_{MSO}^1 \Delta_2$  for some  $t'$ . By Lemma 5,  $\mathcal{S} \lesssim_{FO} t' \lesssim_{FO} \Delta_2$ . Hence  $\mathcal{S} \lesssim_{FO} \Delta_2$ . The converse is obvious.  $\square$

A similar approach can be used for characterising the Caucal hierarchy [8], i.e a form of extension of prefix-recognisable graphs to ‘higher-order’. We use here the characterisation in [6] as a definition:

- The structures in  $Struct_0$  are the finite structures.
- The graphs in  $Graph_n$  are the graphs<sup>2</sup> in  $Struct_n$ .
- The trees in  $Tree_{n+1}$  are the unfolding of graphs in  $Graph_n$ .
- A structure  $\mathcal{S}$  is in  $Struct_{n+1}$  if  $\mathcal{S} \lesssim_{MSO} t$  for some tree  $t$  in  $Tree_{n+1}$ .

The following theorem shows that in the definition of this hierarchy, the monadic logic can be replaced by first-order logic.

**Theorem 5.**  $\mathcal{S}$  is in  $Struct_{n+1}$  iff  $\mathcal{S} \lesssim_{FO} t$  for some tree  $t$  in  $Tree_{n+1}$ .

In fact, this is a direct combination of the definitions, of Theorem [2], and of the following proposition (see [6], Proposition 1).

**Proposition 1.** For  $t$  in  $Tree_n$ , every  $t' \lesssim_{MSO}^1 t$  is also in  $Tree_n$ .

## 4 Other Consequences, Perspectives

### Determinisation of Regular Languages of Infinite Words

From our result can also be derived McNaughton’s determinisation theorem [13]. This theorem states that for every regular language of infinite words of length  $\omega$  (regular meaning accepted by a Büchi automaton), there exists a *deterministic parity automaton* which accepts the same language. Such an automaton is a standard finite deterministic automaton without final states, the states of which are labelled by natural numbers among  $1, \dots, p$  called their *priorities*. An infinite word is accepted by such an automaton if the least priority appearing infinitely often in the (unique by determinism) run is even. The deteterminisation result is fundamental in the theory of languages of infinite words. In particular, it is used in most proofs of the theorem of Rabin.

Let us sketch the link with our result. Given a Büchi automaton accepting a language  $L$ , it is natural to associate to it a structure of finite semigroup  $S$  as well as a morphism  $\varphi$  from finite words to  $S$  such that it is sufficient to know  $\varphi(u_1), \varphi(u_2), \dots$  for determining whether the word  $u_1u_2\dots$  belongs to  $L$  (this was already the approach of Büchi in his proof of complementation [4], see also [15] for the more explicit presentation via  $\omega$ -semigroups). In particular, given two words  $u, v$ , the membership of  $uv^\omega = uvvv\dots$  in  $L$  does only depend of  $\varphi(u)$  and  $\varphi(v)$ .

One can apply Theorem [1] to this semigroup, and obtain for each word  $u$  the forward Ramseyan split  $s_u$ . One constructs a deterministic parity automaton which, when reading a word of  $u$  of length  $n$ , reaches a state of priority:

$$\begin{cases} 2s_u(n) & \text{if } u_{0,m}u_{m,n}^\omega \in L \text{ for } m = \max\{m < n : s_u(n) = s_u(m)\}, \\ 2s_u(n) + 1 & \text{if } u_{0,m}u_{m,n}^\omega \notin L, \text{ or } m \text{ does not exist.} \end{cases}$$

---

<sup>2</sup> In the original version, graphs have their edges labelled. We drop this since it has no impact on the definition of  $Struct_n$ .

One checks that a) this can be implemented by a finite deterministic parity automaton, and b) that the language it accepts is  $L$  (using forward Ramseyanity).

This construction yields a doubly exponential automaton in the size of the original automaton. It is comparable in complexity to the original proof of McNaughton, but quiet far from Safra's construction [18]. A clother study of the construction above shows that in fact only a third of the proof of Theorem 1 (the proof treats separately three different cases) is sufficient for establishing McNaughton's determinisation result. This means that determinisation does not illustrate the full interest of Theorem 1 and should be considered more as a side-effect.

Another combinatorial approach to determinisation of Büchi automata is known from [22]. The combinatorial lemma it involves is suitable for determinisation, but is not as expressive as Theorem 1. And in particular it is not sufficient for proving Theorem 2.

## Infinite Trees

Another motivation for factorising trees is the perspective to give a new proof to the theorem of Rabin of decidability of monadic second-order logic over infinite trees [17]. Such a contribution would be an answer to a long lasting open question of Shelah [19]. It would also generalise the original proof of Büchi for infinite words to the case of infinite trees. Theorem 1 is far from being sufficient for such an application. A reason for that is the irreducibility of nondeterminism in tree automata (it is for instance shown in [14] that some languages of infinite trees are not accepted by any unambiguous automaton, i.e., an automaton having a single accepting run for each accepted input tree). Standard proofs of the result of Rabin handle this problem using game theory, see e.g., [23]. The theorem developed in this paper has no nondeterminism feature, and for this reason cannot be sufficient alone for this application.

Despite this, the main reason for the introduction of Theorem 1 by the author is its perspective of usage in an extension of the work in [2] to infinite trees, which would be at the same time an extension of Rabin's theorem

## Acknowledgement

Many thanks to Achim Blumensath who follows this work from the beginning.

## References

1. Blumensath, A.: Prefix-recognisable graphs and monadic second-order logic. Technical Report AIB-06-2001, RWTH Aachen (May 2001)
2. Bojańczyk, M., Colcombet, T.: Bounds in omega-regularity. In: IEEE Symposium on Logic In Computer Science, pp. 285–296. IEEE Computer Society Press, Los Alamitos (2006)
3. Brown, T.C.: An interesting combinatorial method in the theory of locally finite semigroups. *Pacific Journal of Mathematics* 36(2), 277–294 (1971)



4. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Proceedings of the International Congress on Logic, Methodology and Philosophy of Science, pp. 1–11. Stanford University press, Stanford (1960)
5. Büchi, J.R., Elgot, C.C.: Decision problems of weak second order arithmetics and finite automata. Notices of the American Math. Soc. 5, 834 (1958)
6. Carayol, A., Wöhrle, S.: The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In: Pandya, P.K., Radhakrishnan, J. (eds.) FST TCS 2003. LNCS, vol. 2914, pp. 112–123. Springer, Heidelberg (2003)
7. Caucal, D.: On infinite transition graphs having a decidable monadic theory. In: Meyer auf der Heide, F., Monien, B. (eds.) ICALP 1996. LNCS, vol. 1099, pp. 194–205. Springer, Heidelberg (1996)
8. Caucal, D.: On infinite terms having a decidable monadic theory. In: Diks, K., Rytter, W. (eds.) MFCS 2002. LNCS, vol. 2420, pp. 165–176. Springer, Heidelberg (2002)
9. Chalopin, J., Leung, H.: On factorization forests of finite height. Theoretical Computer Science 310(1–3), 489–499 (2004)
10. Green, J.A.: On the structure of semigroups. Annals of Mathematics 54(1), 163–172 (1951)
11. Hashiguchi, K.: Relative star height, star height and finite automata with distance functions. In: Formal Properties of Finite Automata and Applications, pp. 74–88 (1988)
12. Kirsten, D.: Distance desert automata and the star height problem. RAIRO 3(39), 455–509 (2005)
13. McNaughton, R.: Testing and generating infinite sequences by a finite automaton. Information and Control 9(5), 521–530 (1966)
14. Niwinski, D., Walukiewicz, I.: Ambiguity problem for automata on infinite trees. Personal communication (1997)
15. Perrin, D., Pin, J-E.: Semigroups, Formal Languages and Groups. In: chapter Semigroups and automata on infinite words, pp. 49–72. Kluwer Academic Publishers, Dordrecht (1995)
16. Pin, J-E., Weil, P.: Polynomial closure and unambiguous product. Theory Comput. Syst. 30(4), 383–422 (1997)
17. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Trans. Amer. Math. soc. 141, 1–35 (1969)
18. Safra, S.: On the complexity of omega-automata. In: FOCS, pp. 319–327 (1988)
19. Shelah, S.: The monadic theory of order. Annals Math. 102, 379–419 (1975)
20. Simon, I.: Factorization forests of finite height. Theor. Comput. Sci. 72(1), 65–94 (1990)
21. Simon, I.: On semigroups of matrices over the tropical semiring. ITA 28(3-4), 277–294 (1994)
22. Thomas, W.: A combinatorial approach to the theory of  $\omega$ -automata. Information and Control 48, 261–283 (1981)
23. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Language Theory, vol. III, pp. 389–455. Springer, Heidelberg (1997)

# Model Theory Makes Formulas Large

Anuj Dawar<sup>1</sup>, Martin Grohe<sup>2</sup>, Stephan Kreutzer<sup>2</sup>, and Nicole Schweikardt<sup>2</sup>

<sup>1</sup> University of Cambridge, U.K

anuj.dawar@cl.cam.ac.uk

<sup>2</sup> Institut für Informatik, Humboldt Universität zu Berlin

{grohe,kreutzer,schweika}@informatik.hu-berlin.de

**Abstract.** Gaifman’s locality theorem states that every first-order sentence is equivalent to a local sentence. We show that there is no elementary bound on the length of the local sentence in terms of the original.

The classical Łoś-Tarski theorem states that every first-order sentence preserved under extensions is equivalent to an existential sentence. We show that there is no elementary bound on the length of the existential sentence in terms of the original. Recently, variants of the Łoś-Tarski theorem have been proved for certain classes of finite structures, among them the class of finite acyclic structures and more generally classes of structures of bounded tree width. Our lower bound also applies to these variants.

We further prove that a version of the Feferman-Vaught theorem based on a restriction by formula length necessarily entails a non-elementary blow-up in formula size.

All these results are based on a similar technique of encoding large numbers by trees of small height in such a way that small formulas can speak about these numbers. Notably, our lower bounds do not apply to restrictions of the results to structures of bounded degree. For such structures, we obtain elementary upper bounds in all cases. However, even there we can prove at least doubly exponential lower bounds.

## 1 Introduction

Classical results of model theory provide syntactical normal forms for various semantical properties of structures. For example, the Łoś-Tarski theorem states that every first-order definable property that is preserved under extensions of structures is actually definable by an existential first-order sentence. Gaifman’s locality theorem provides a normal form for all properties definable in first-order logic. It states that each first-order definable property is definable by a local sentence, that is, a sentence where quantification is basically restricted to local neighbourhoods of elements.

Gaifman’s theorem has found various applications in algorithms and complexity [9, 4, 16, 17]. In particular, there are very general algorithmic meta-theorems stating that first-order model-checking is fixed-parameter tractable for various classes of structures, such as planar graphs or graphs with excluded minors, and that first-order definable optimisation problems on such classes have polynomial time approximation schemes. These algorithms are heavily based on (an effective version of) Gaifman’s theorem:

First-order formulas are first translated into local formulas, and then these local formulas are algorithmically evaluated.

While it is known that the Łoś-Tarski theorem fails when restricted to all finite structures, it has recently been proved [1] that the theorem does still hold when restricted to specific “well-behaved” classes of finite structures such as acyclic structures, structures of bounded tree width, and structures of bounded degree.

In the context of algorithms, complexity, and finite model theory, questions about the efficiency of the normal forms, which are usually neglected in classical model theory, are of fundamental importance. These are the questions we address. By efficiency we mean the size of the formulas in normal form (*succinctness*) and the existence of efficient algorithms that translate formulas into their normal forms (*complexity of the translation*). We shall prove nonelementary lower bounds for the succinctness — obviously, this implies nonelementary lower bounds on the complexity of the translation. Specifically, we prove that there is no elementary function  $f$  such that every first-order sentence  $\varphi$  is equivalent to a local first-order sentence  $\tilde{\varphi}$  of length  $\|\tilde{\varphi}\| \leq f(\|\varphi\|)$ , not even on the class of all finite trees. Similarly, we prove that there is no elementary function  $f$  such that every first-order sentence  $\varphi$  that is preserved under extensions (on arbitrary structures) is equivalent to an existential first-order sentence  $\tilde{\varphi}$  of length  $\|\tilde{\varphi}\| \leq f(\|\varphi\|)$ , not even on the class of all finite trees. This provides a succinctness lower bound for both the classical Łoś-Tarski theorem and its variants for classes of finite forests and all classes of finite structures that contain all trees (but not for classes of finite structures of bounded degree).

We prove a further lower bound that is concerned with the classical Feferman-Vaught theorem. The classical theorem states that for certain forms of compositions of structures the theory of a structure composed from simpler structures is determined by the theories of the simpler structures. In particular, there is a function  $f$  such that if structures  $\mathcal{A}_i$  and  $\mathcal{B}_i$  (for  $i = 1, 2$ ) satisfy the same first-order sentences of length at most  $f(\ell)$ , then the disjoint union of  $\mathcal{A}_1$  and  $\mathcal{A}_2$  satisfies the same first-order sentences of size  $\ell$  as the disjoint union of  $\mathcal{B}_1$  and  $\mathcal{B}_2$ . We prove a lower bound on the growth rate of  $f$  showing that  $f$  is not bounded above by an elementary function.

Technically, all our lower bound proofs rely on a suitable encoding of large natural numbers by trees of small height that can be controlled by small first-order formulas. In fact, we show — and use — that full arithmetic on a large initial segment of the positive integers can be simulated by comparably small first-order formulas that operate on the tree encodings of the numbers. It is worth mentioning that this approach can be applied in various other contexts. For example, concerning the classical decision problem, it is known that the first-order theory (and actually also the monadic second-order theory) of trees is decidable [23, 19]; and in [2] (see also [7] for related results) it has been shown that there is no *elementary* decision algorithm. A simple proof of this non-elementary lower bound can easily be obtained using the methods in the present paper (details of this can be found in the full version of this paper).

A point to note, however, is that all our non-elementary lower bounds heavily rely on the fact that the degree of the underlying structures is unbounded. In fact, when restricting attention to classes of structures of bounded degree, we can show elementary upper bounds as counterparts of the non-elementary lower bounds on classes of structures of

unbounded degree. In particular, in the bounded degree case we obtain a 4-fold exponential upper bound for Gaifman’s locality theorem; and we get a 5-fold exponential upper bound for the variant of the Łoś-Tarski theorem on the class of acyclic structures of bounded degree.

As far as we know, techniques similar to those applied here go back to Stockmeyer and Meyer [21]. Much later, such techniques have been employed in [10, 18, 12, 13] to prove lower bounds in parameterised complexity, respectively, on the succinctness of monadic logics. A related succinctness lower bound deserves mention. It has recently been proved by Rossman [20] that the homomorphism preservation theorem (in contrast with the Łoś-Tarski theorem) holds in the class of all finite structures. Here, it is known that there is no elementary bound on the length of the existential positive formula obtained.<sup>1</sup>

The rest of the paper is structured as follows. Section 2 establishes some definitions and notation and Section 3 presents the encoding of numbers by trees that is then used to prove lower bounds on the size of formulas in Gaifman normal form (Section 4) and the failure of the Feferman-Vaught theorem for formula length (Section 5). Section 6 then establishes the lower bound for the Łoś-Tarski theorem, which is based on a different encoding of numbers by trees. Finally, Section 7 contains the elementary upper bounds on classes of structures of bounded degree. Due to space limitations, many technical details of the proofs are deferred to the full version of this paper.

## 2 Preliminaries

We use  $\mathbb{R}$  to denote the set of reals and  $\mathbb{N}$  to denote the set of natural numbers, i.e., the set of nonnegative integers. For natural numbers  $m < n$  we write  $[m, n]$  to denote the set  $\{m, m+1, \dots, n\}$ .

We say that a function  $f : \mathbb{N} \rightarrow \mathbb{R}$  is (*1-fold*) *exponential* if there is some polynomial  $p$  such that  $f(n)$  is eventually bounded by  $2^{p(n)}$ . For any  $k \geq 2$ , a function  $f$  is called *k-fold exponential* if there is some  $(k-1)$ -fold exponential function  $g$  such that  $f(n)$  is eventually bounded by  $2^{g(n)}$ . A function  $f : \mathbb{N} \rightarrow \mathbb{R}$  is called *elementary* if it can be formed from the successor function, addition, subtraction, and multiplication using compositions, projections, bounded additions, and bounded multiplications (of the form  $\sum_{z \leq y} g(\bar{x}, z)$  and  $\prod_{z \leq y} g(\bar{x}, z)$ ). The crucial fact for us is that a function  $f$  is bounded by an elementary function if, and only if, there exists a  $k \geq 1$  such that  $f$  is bounded by a  $k$ -fold exponential function (see, e.g., [3]).

One function of particular interest for the present paper is the function  $Tower : \mathbb{N} \rightarrow \mathbb{N}$ , defined via  $Tower(0) := 1$  and, for all  $h \geq 1$ ,  $Tower(h) := 2^{Tower(h-1)}$ . I.e.,  $Tower(h)$  is a tower of 2s of height  $h$ . Note that, e.g., none of the functions  $Tower(h)$ ,  $Tower(\sqrt[4]{h})$ ,  $Tower(\log h)$  is bounded by an elementary function.

A *vocabulary* is a finite set of relation symbols and constant symbols. Associated with every relation symbol  $R$  is a positive integer called the *arity* of  $R$ . In the following,  $\tau$  always denotes a vocabulary. A vocabulary  $\tau$  is called *relational* if it does not contain any constant symbol. A  $\tau$ -*structure*  $\mathcal{A}$  consists of a non-empty set  $A$ , called the *universe*

<sup>1</sup> This is mentioned by Rossman citing unpublished work of Gurevich and Shelah. As far as we are aware, a proof of this lower bound has not yet been published.

of  $\mathcal{A}$ , an element  $c^{\mathcal{A}} \in A$  for each constant symbol  $c \in \tau$ , and a relation  $R^{\mathcal{A}} \subseteq A^r$  for each  $r$ -ary relation symbol  $R \in \tau$ .  $\mathcal{A}$  is called an *induced substructure* of a  $\tau$ -structure  $\mathcal{B}$  if  $A \subseteq B$ ,  $R^{\mathcal{A}} = R^{\mathcal{B}} \cap A^r$ , for each relation symbol  $R \in \tau$  of arity  $r$ , and  $c^{\mathcal{A}} = c^{\mathcal{B}}$  for each constant symbol  $c \in \tau$ .

The *Gaifman graph* of a  $\tau$ -structure  $\mathcal{A}$  is the (undirected, loop-free) graph  $\mathcal{G}_{\mathcal{A}}$  with vertex set  $A$  and an edge between two vertices  $a, b \in A$  iff there exists an  $R \in \tau$  and a tuple  $(a_1, \dots, a_r) \in R^{\mathcal{A}}$  such that  $a, b \in \{a_1, \dots, a_r\}$ . The *distance* between two elements  $a, b \in A$  in  $\mathcal{A}$ , denoted by  $dist^{\mathcal{A}}(a, b)$ , is defined to be the length (that is, number of edges) of the shortest path from  $a$  to  $b$  in the Gaifman graph of  $\mathcal{A}$ . For  $r \geq 0$  and  $a \in A$ , the  $r$ -neighbourhood of  $a$  in  $\mathcal{A}$  is the set  $N_r^{\mathcal{A}}(a) = \{b \in A : dist^{\mathcal{A}}(a, b) \leq r\}$ . The induced substructure of  $\mathcal{A}$  with universe  $N_r^{\mathcal{A}}(a)$  is denoted by  $\mathcal{N}_r^{\mathcal{A}}(a)$ . We omit superscripts  $\mathcal{A}$  if  $\mathcal{A}$  is clear from the context.

We write  $FO(\tau)$  to denote the class of all formulae of first-order logic over the vocabulary  $\tau$ , and we write  $qr(\varphi)$  to denote the *quantifier rank* of an  $FO(\tau)$ -formula  $\varphi$ . In a natural way, we view formulas as trees (to be precise, as their *syntax trees*), where leaves correspond to the atoms of the formulas, and inner vertices correspond to Boolean connectives or quantifiers. We define the *size* (or, *length*)  $\|\varphi\|$  of a first-order formula  $\varphi$  as the number of vertices in the syntax tree of  $\varphi$ .

Whenever we write  $E$ , it denotes a binary relation symbol. We view  $\{E\}$ -structures as directed graphs. For a directed graph  $\mathcal{A} = (A, E^{\mathcal{A}})$  and an  $a \in A$ , we let  $A_a$  be the set of all vertices  $b$  such there is a path from  $a$  to  $b$  (this includes  $a$ ), and we let  $\mathcal{A}_a$  be the induced substructure of  $\mathcal{A}$  with universe  $A_a$ . Unless we explicitly call them *undirected*, we view trees as being directed from the root to the leaves. A *forest* is a directed graph in which every vertex has indegree at most 1. Vertices of indegree 0 are called *roots* of the forest. A *tree* is a forest with exactly one root. The class of all finite forests is denoted by  $\mathfrak{F}$  and the class of all finite trees by  $\mathfrak{T}$ . The *height* of a tree  $\mathcal{T}$  is the length of the longest path in  $\mathcal{T}$ .

### 3 Encoding Numbers by Trees

In this section we introduce the technical machinery that is used for proving our main theorems in sections 4 and 5. We use the following encoding of natural numbers by trees, introduced in [8].

**Definition 1 (Encoding numbers by trees).** For  $i, n \in \mathbb{N}$  we write  $bit(i, n)$  to denote the  $i$ -th bit in the binary representation of  $n$ . I.e.,  $bit(i, n) = 0$  if  $\lfloor \frac{n}{2^i} \rfloor$  is even, and  $bit(i, n) = 1$  if  $\lfloor \frac{n}{2^i} \rfloor$  is odd. Inductively we define a tree  $\mathcal{T}(n)$  for each natural number  $n$  as follows:  $\mathcal{T}(0)$  is the one-node tree. For  $n \geq 1$  the tree  $\mathcal{T}(n)$  is obtained by creating a new root and attaching to it all trees  $\mathcal{T}(i)$  for all  $i$  such that  $bit(i, n) = 1$ .

Illustrations of these trees can be found in [8]. It is straightforward to see (cf. [8, Lemma 10.20]) that for all  $h, n \geq 0$ ,  $height(\mathcal{T}(n)) \leq h \iff n < Tower(h)$ . The next lemma from [8] shows that the tree encodings of numbers can be “controlled” by small first-order formulas. (In [8], the statement of the lemma is formulated for trees instead of general structures. The proof given there, however, also holds for general structures and thus leads to the following lemma.)

**Lemma 1** ([8, Lemma 10.21]). *For every  $h \geq 0$  there is an  $\text{FO}(E)$ -formula  $\text{eq}_h(x, y)$  of length  $\mathcal{O}(h)$  such that for all structures  $\mathcal{A} = (A, E^{\mathcal{A}})$  and  $t, u \in A$  we have: If there are  $m, n < \text{Tower}(h)$  such that the structures  $\mathcal{A}_t$  and  $\mathcal{A}_u$  are isomorphic to  $\mathcal{T}(m)$  and  $\mathcal{T}(n)$ , resp., then  $\mathcal{A} \models \text{eq}_h(t, u) \iff m = n$ .*

Using this, one easily obtains the following two lemmas which provide formulas of length polynomial in  $h$  that recognise tree encodings and define arithmetic on the tree encodings of numbers of size up to  $\text{Tower}(h)$ .

**Lemma 2.** *For every  $h \geq 0$  there is a  $\text{FO}(E)$ -formula  $\text{encoding}_h(x)$  of length  $\mathcal{O}(h^2)$  such that for all structures  $\mathcal{A} = (T, E^{\mathcal{A}})$  and  $t \in A$  we have:  $\mathcal{A} \models \text{encoding}_h(t) \iff$  there is an  $i \in \{0, \dots, \text{Tower}(h)-1\}$  such that  $\mathcal{A}_t$  is isomorphic to  $\mathcal{T}(i)$ .*

**Lemma 3.** *For every  $h \geq 0$  there are  $\text{FO}(E)$ -formulas  $\text{bit}_h(x, y)$  of size  $\mathcal{O}(h)$ ,  $\text{less}_h(x, y)$  of size  $\mathcal{O}(h^2)$ ,  $\text{min}(x)$  of constant size (not depending on  $h$ ),  $\text{succ}_h(x, y)$  of size  $\mathcal{O}(h^3)$ , and  $\text{max}_h(x)$  of size  $\mathcal{O}(h^4)$  such that for all structures  $\mathcal{A} = (A, E^{\mathcal{A}})$  and  $t, u \in A$  we have: If there are  $m, n < \text{Tower}(h)$  such that the structures  $\mathcal{A}_t$  and  $\mathcal{A}_u$  are isomorphic to  $\mathcal{T}(m)$  and  $\mathcal{T}(n)$ , respectively, then we have:*

- (a)  $\mathcal{A} \models \text{bit}_h(t, u) \iff \text{bit}(m, n) = 1$
- (b)  $\mathcal{A} \models \text{less}_h(t, u) \iff m < n$
- (c)  $\mathcal{A} \models \text{min}(t) \iff \mathcal{A}_t$  is isomorphic to  $\mathcal{T}(0)$
- (d)  $\mathcal{A} \models \text{succ}_h(t, u) \iff m + 1 = n$
- (e)  $\mathcal{A} \models \text{max}_h(t) \iff \mathcal{A}_t$  is isomorphic to  $\mathcal{T}(\text{Tower}(h)-1)$ .

## 4 Lower Bounds for the Size of Formulas in Gaifman Normal Form

The aim of this section is to prove a non-elementary succinctness gap for Gaifman’s theorem. To give a precise formulation of Gaifman’s theorem and our new bounds on formula length, we need to fix some (standard) notation.

For every  $r \geq 0$ , we let  $\text{dist}_{\leq r}(x, y)$  be an  $\text{FO}(\tau)$ -formula expressing that the distance between  $x$  and  $y$  is at most  $r$ . We often write  $\text{dist}(x, y) \leq r$  instead of  $\text{dist}_{\leq r}(x, y)$  and  $\text{dist}(x, y) > r$  or  $\text{dist}_{> r}(x, y)$  instead of  $\neg \text{dist}_{\leq r}(x, y)$ . An  $\text{FO}(\tau)$ -formula  $\psi(x)$  is called  $r$ -local if for every  $\tau$ -structure  $\mathcal{A}$  and every  $a \in A$  we have  $\mathcal{A} \models \psi(a) \iff \mathcal{N}_r^{\mathcal{A}}(a) \models \psi(a)$ . A *basic local sentence* (with parameters  $k, r$ ) is a sentence of the form

$$\exists x_1 \cdots \exists x_k \left( \bigwedge_{1 \leq i < j \leq k} \text{dist}(x_i, x_j) > 2r \wedge \bigwedge_{1 \leq i \leq k} \psi(x_i) \right),$$

where  $\psi(x)$  is  $r$ -local.

For an  $\text{FO}(\tau)$ -sentence  $\varphi$  we say that  $\varphi$  is in *Gaifman normal form* if  $\varphi$  is a Boolean combination of basic local sentences. Gaifman’s well-known theorem [11] states that every first-order sentence over a relational vocabulary is equivalent to a first-order sentence in Gaifman normal form. The proof in [11] proceeds by induction on the length of the given first-order sentence  $\varphi$  and leads to an effective algorithm that transforms a

given  $\varphi$  into an equivalent sentence  $\psi$  in Gaifman normal form. A closer look at Gaifman’s proof shows that the size of the constructed sentence  $\psi$  may be non-elementary in the size of the original sentence  $\varphi$ . The main result of the present section shows that this huge increase in formula size is not just an artifact of Gaifman’s proof, but that indeed there are first-order formulas  $\varphi$  for which the shortest equivalent formula in Gaifman normal form is non-elementarily larger than  $\varphi$ .

**Theorem 1.** *For every  $h \geq 1$  there is an FO( $E$ )-sentence  $\varphi_h$  of size  $\mathcal{O}(h^4)$  such that every FO( $E$ )-sentence in Gaifman normal form that is equivalent to  $\varphi_h$  on the class  $\mathfrak{T}$  of finite trees has size at least  $\text{Tower}(h)$ .*

Here, we show the following variant that speaks about the class of all *forests* rather than *trees*. The proof of Theorem 2 avoids some of the unpleasant details needed in the proof of Theorem 1 while still exposing the main ideas that are crucial for the proof of Theorem 1. The proof of Theorem 1 can be found in the full version of this paper.

**Theorem 2.** *For every  $h \geq 1$  there is an FO( $E$ )-sentence  $\varphi_h$  of size  $\mathcal{O}(h^4)$  such that every FO( $E$ )-sentence in Gaifman normal form that is equivalent to  $\varphi_h$  on the class  $\mathfrak{F}_{\leq h}$  of finite forests of height  $\leq h$  has size at least  $\text{Tower}(h)$ .*

*Proof.* We use the tree encodings of natural numbers introduced in Section 3. For  $h \geq 1$  we define the structure  $\mathcal{F}_h$  to be the forest that consists of the disjoint union of all trees  $\mathcal{T}(j)$  for all  $j \in \{0, \dots, \text{Tower}(h) - 1\}$ . Furthermore, for every  $i \in \{0, \dots, \text{Tower}(h) - 1\}$ , we let  $\mathcal{F}_h^{-i}$  be the forest that consists of the disjoint union of all trees  $\mathcal{T}(j)$  for all  $j$  with  $j \neq i$  and  $j \in \{0, \dots, \text{Tower}(h) - 1\}$ .

We let  $\text{root}(x)$  be a formula which expresses that a node  $x$  has in-degree 0, i.e.,  $\text{root}(x) := \neg \exists y E(y, x)$ . We choose the FO( $E$ )-sentence  $\varphi_h$  as follows:  $\varphi_h :=$

$$\exists x (\text{root}(x) \wedge \text{min}(x)) \wedge \forall y \left( (\text{root}(y) \wedge \neg \text{max}_h(y)) \rightarrow \exists z (\text{root}(z) \wedge \text{succ}_h(y, z)) \right).$$

Using Lemma 3 it is straightforward to see that  $\|\varphi_h\| = \mathcal{O}(h^4)$  and

$$\mathcal{F}_h \models \varphi_h \quad \text{and, for each } i < \text{Tower}(h), \quad \mathcal{F}_h^{-i} \not\models \varphi_h. \tag{1}$$

Now let  $\psi$  be an FO( $E$ )-sentence in Gaifman normal form that is equivalent to  $\varphi_h$  on the class  $\mathfrak{F}_{\leq h}$ . In particular, since  $\mathcal{F}_h$  as well as all the  $\mathcal{F}_h^{-i}$  belong to  $\mathfrak{F}_{\leq h}$ , we obtain from (1) that

$$\mathcal{F}_h \models \psi \quad \text{and, for each } i < \text{Tower}(h), \quad \mathcal{F}_h^{-i} \not\models \psi. \tag{2}$$

Our aim is to show that  $H := \|\psi\| \geq \text{Tower}(h)$ . Aiming at a contradiction, let us now assume that  $H < \text{Tower}(h)$ .

Since  $\psi$  is in Gaifman normal form, it is a Boolean combination of *basic local sentences*  $\chi_1, \dots, \chi_L$ , where each  $\chi_\ell$  (for  $\ell \in \{1, \dots, L\}$ ) is of the form

$$\chi_\ell := \exists x_1 \dots \exists x_{k_\ell} \left( \bigwedge_{1 \leq i < j \leq k_\ell} \text{dist}(x_i, x_j) > 2r_\ell \wedge \bigwedge_{1 \leq i \leq k_\ell} \psi_\ell(x_i) \right),$$

with  $k_\ell, r_\ell \geq 1$  and  $\psi_\ell(x)$  a formula that is  $r_\ell$ -local. In particular,

$$H := \|\psi\| \geq k_1 + \dots + k_L. \tag{3}$$

We can assume w.l.o.g. that there exists an  $\tilde{L}$  with  $0 \leq \tilde{L} \leq L$  such that

$$\text{for each } \ell \leq \tilde{L}, \mathcal{F}_h \models \chi_\ell, \quad \text{and} \quad \text{for each } \ell > \tilde{L}, \mathcal{F}_h \not\models \chi_\ell. \tag{4}$$

For all  $\ell \leq \tilde{L}$  we know that  $\mathcal{F}_h \models \chi_\ell$ , i.e., there are nodes  $t_1^{(\ell)}, \dots, t_{k_\ell}^{(\ell)}$  in  $\mathcal{F}_h$  such that the formula

$$\bigwedge_{1 \leq i < j \leq k_\ell} \text{dist}(x_i, x_j) > 2r_\ell \wedge \bigwedge_{1 \leq i \leq k_\ell} \psi_\ell(x_i) \tag{5}$$

is satisfied in  $\mathcal{F}_h$  when interpreting each variable  $x_i$  with the node  $t_i^{(\ell)}$ . The set  $\{t_i^{(\ell)} : \ell \leq \tilde{L} \text{ and } i \leq k_\ell\}$  consists of at most  $k_1 + \dots + k_{\tilde{L}} \leq H$  nodes (see (3)). Since we assume that  $H < \text{Tower}(h)$ , and since  $\mathcal{F}_h$  consists of  $\text{Tower}(h)$  disjoint trees, there must be at least one component  $\mathcal{T}$  of  $\mathcal{F}_h$  in which none of the nodes from  $\{t_i^{(\ell)} : \ell \leq \tilde{L} \text{ and } i \leq k_\ell\}$  is present. Let  $j \in \{0, \dots, \text{Tower}(h)-1\}$  be such that  $\mathcal{T} = \mathcal{T}(j)$ .

Now, of course, the forest  $\mathcal{F}_h^{-j}$ , which is obtained from  $\mathcal{F}_h$  by removing the component  $\mathcal{T}(j)$ , still contains all the nodes in  $\{t_i^{(\ell)} : \ell \leq \tilde{L} \text{ and } i \leq k_\ell\}$ . Considering (5), note that each formula  $\psi_\ell(x_i)$  is  $r_\ell$ -local around  $x_i$ . Thus, when interpreting  $x_i$  with the node  $t_i^{(\ell)}$ , the formula can only “speak” about the  $r_\ell$ -neighbourhood of  $t_i^{(\ell)}$ , which is the same in  $\mathcal{F}_h^{-j}$  as in  $\mathcal{F}_h$ . We thus obtain from (5) that  $\mathcal{F}_h^{-j} \models \chi_\ell$  for each  $\ell \leq \tilde{L}$ .

Let us now consider the formulas  $\chi_\ell$  with  $\ell > \tilde{L}$ . From (4) we know that  $\mathcal{F}_h \not\models \chi_\ell$ , i.e.,  $\mathcal{F}_h \models \neg\chi_\ell$ , where the formula  $\neg\chi_\ell$  is of the following form:

$$\neg\exists x_1 \dots \exists x_{k_\ell} \left( \bigwedge_{1 \leq i < j \leq k_\ell} \text{dist}(x_i, x_j) > 2r_\ell \wedge \bigwedge_{1 \leq i \leq k_\ell} \psi_\ell(x_i) \right).$$

Since the formula  $\psi_\ell(x_i)$  is  $r_\ell$ -local and since  $\mathcal{F}_h^{-j}$  is obtained from  $\mathcal{F}_h$  by removing an entire component of  $\mathcal{F}_h$ , it is straightforward to see that also  $\mathcal{F}_h^{-j} \models \neg\chi_\ell$ . In total, we now know the following:

$$\text{for each } \ell \leq \tilde{L}, \mathcal{F}_h^{-j} \models \chi_\ell, \quad \text{and} \quad \text{for each } \ell > \tilde{L}, \mathcal{F}_h^{-j} \not\models \chi_\ell. \tag{6}$$

From (6) and (4) we obtain that  $\mathcal{F}_h^{-j}$  satisfies exactly the same basic local sentences from  $\{\chi_1, \dots, \chi_L\}$  as  $\mathcal{F}_h$ . Since  $\psi$  is a Boolean combination of the sentences  $\chi_1, \dots, \chi_L$ , we thus have that  $\mathcal{F}_h^{-j} \models \psi \iff \mathcal{F}_h \models \psi$ . This, however, is a contradiction to (2). Altogether, the proof of Theorem 2 is complete.  $\square$

To conclude this section let us mention that an easy reduction shows that Theorem 1 and Theorem 2 still hold when replacing  $\mathfrak{T}$  and  $\mathfrak{F}_{\leq h}$  by the classes  $\mathfrak{T}^u$  and  $\mathfrak{F}_{\leq h}^u$  of *undirected* trees and *undirected* forests of height at most  $h$ , respectively.

### 5 Failure of Feferman-Vaught Theorems for Formula Size

The classical Feferman-Vaught theorem [6] states that for certain forms of compositions of structures the theory of a structure composed from simpler structures is determined



by the theories of the simpler structures. The plainest form of composition is the *disjoint union*, denoted by  $\oplus$  in the following. The Feferman-Vaught theorem for disjoint union and first-order logic states that for all structures  $\mathcal{A}_1, \mathcal{A}_2, \mathcal{B}_1, \mathcal{B}_2$ , if the structures  $\mathcal{A}_i$  and  $\mathcal{B}_i$  (for  $i = 1, 2$ ) satisfy the same first-order sentences, their disjoint unions  $\mathcal{A}_1 \oplus \mathcal{A}_2$  and  $\mathcal{B}_1 \oplus \mathcal{B}_2$  also satisfy the same first-order sentences. This can be stratified by the quantifier rank, that is, if  $\mathcal{A}_i$  and  $\mathcal{B}_i$  satisfy the same first-order sentences of quantifier rank at most  $q$ , then  $\mathcal{A}_1 \oplus \mathcal{A}_2$  and  $\mathcal{B}_1 \oplus \mathcal{B}_2$  also satisfy the same first-order sentences of quantifier rank at most  $q$ . This result is an immensely useful tool in analysing the expressivity of first order logic, and for deriving bounds on the quantifier rank.

To derive bounds on the formula size, it would be similarly useful to have an analogous result for formula size instead of quantifier rank. As (for a fixed, finite vocabulary) there are only finitely many first-order sentences of quantifier rank  $q$ , up to logical equivalence, we immediately get the following: There is a function  $f$  such that if the structures  $\mathcal{A}_i$  and  $\mathcal{B}_i$  (for  $i = 1, 2$ ) satisfy the same first-order sentences of length at most  $f(\ell)$ , then  $\mathcal{A}_1 \oplus \mathcal{A}_2$  and  $\mathcal{B}_1 \oplus \mathcal{B}_2$  satisfy the same first-order sentences of length at most  $\ell$ . It is not hard to derive an upper bound of  $Tower(\mathcal{O}(\ell))$  on the function  $f$ . Maybe surprisingly, this upper bound is essentially tight:

**Theorem 3.** *There is no elementary function  $f$  such that the following holds for all trees  $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathfrak{T}$ : If  $\mathcal{A}$  and  $\mathcal{B}$  satisfy the same first-order sentences of length at most  $f(\ell)$ , then  $\mathcal{A} \oplus \mathcal{C}$  and  $\mathcal{B} \oplus \mathcal{C}$  satisfy the same first-order sentences of length at most  $\ell$ .*

*Proof.* We use the encoding and the formulas from Section 3. For every  $h \geq 1$ , let

$$\varphi_h := \forall x(\text{encoding}_h(x) \rightarrow (\max_h(x) \vee \exists y \text{succ}_h(x, y))).$$

Then there is a constant  $c \geq 1$  such that  $\|\varphi_h\| \leq c \cdot h^4$  for all  $h$ .

Suppose for contradiction that  $f$  is an elementary function with the desired property. We may assume that  $f(\ell) \geq \ell$  for all  $\ell \geq 1$ . As there are only exponentially many first-order sentences  $\varphi$  of a given length, there is an  $h \geq 1$  such that there are less than  $Tower(h-1)$  first-order sentences of length at most  $f(c \cdot h^4)$  (up to equivalence). Let us fix such an  $h$ , and let  $\ell = c \cdot h^4$  and  $n = Tower(h) - 1$ . For every  $j \in [0, n]$ , let  $\mathcal{F}_j$  denote the forest consisting of the trees  $\mathcal{T}(j), \dots, \mathcal{T}(n)$ , and let  $\mathcal{U}_j$  be the tree obtained from  $\mathcal{F}_j$  by connecting a new root with the roots of all trees in  $\mathcal{F}_j$ . Then there are numbers  $j, k$  such that  $1 \leq j < k \leq n$ , and the trees  $\mathcal{U}_j$  and  $\mathcal{U}_k$  satisfy the same first-order sentences of length at most  $f(\ell)$ . Observe that

$$\mathcal{F}_j \oplus \mathcal{T}(j-1) \models \varphi_h \quad \text{and} \quad \mathcal{F}_k \oplus \mathcal{T}(j-1) \not\models \varphi_h.$$

Now let  $\mathcal{A} = \mathcal{U}_j$ ,  $\mathcal{B} = \mathcal{U}_k$ , and  $\mathcal{C} = \mathcal{T}(j-1)$ . As the new roots of  $\mathcal{A}, \mathcal{B}$  are not nodes satisfying  $\text{encoding}_h(x)$  (because  $\mathcal{A}$  and  $\mathcal{B}$  are isomorphic to trees  $\mathcal{T}(n_{\mathcal{A}})$  and  $\mathcal{T}(n_{\mathcal{B}})$  with  $n_{\mathcal{A}}, n_{\mathcal{B}} \geq Tower(h)$ ), we have  $\mathcal{A} \oplus \mathcal{C} \models \varphi_h$  and  $\mathcal{B} \oplus \mathcal{C} \not\models \varphi_h$ . Since the length of  $\varphi_h$  is at most  $\ell$  and  $\mathcal{A}, \mathcal{B}$  satisfy the same sentences of length at most  $f(\ell)$ , this is a contradiction.  $\square$

## 6 Existential Preservation on Forests

A structure  $\mathcal{B}$  is called an *extension* of  $\mathcal{A}$  if  $\mathcal{A}$  is an induced substructure of  $\mathcal{B}$ . Let  $\tau$  be a vocabulary and let  $\mathfrak{C}$  be a class of finite  $\tau$ -structures that is closed under induced

substructures. An  $\text{FO}(\tau)$ -sentence  $\varphi$  is *preserved under extensions* on  $\mathfrak{C}$  if the following is true for all structures  $\mathcal{A}, \mathcal{B} \in \mathfrak{C}$ : If  $\mathcal{A} \models \varphi$  and  $\mathcal{B}$  is an extension of  $\mathcal{A}$ , then  $\mathcal{B} \models \varphi$ .

The well-known *Łoś-Tarski Theorem* (see e.g. [15]) states that every first-order sentence that is preserved under extensions on the class of *all* structures (i.e., finite as well as infinite structures), is equivalent to an *existential* first-order sentence. Here, the class of *existential first-order formulas* is obtained by closing the atomic formulas and the negated atomic formulas under conjunction, disjunction, and existential quantification. While the *Łoś-Tarski theorem* fails when shifting the attention from the class of *all* structures to the class of all *finite* structures ([22, 14]), it was shown in [1] that the Łoś-Tarski theorem holds when restricted to certain “well-behaved” classes of finite structures, among them the class of all finite acyclic structures. The main result of the present section, Theorem 4, shows that a translation of formulas preserved under extensions into equivalent existential formulas may increase the formula size non-elementarily.

In the following, we let  $L$  and  $X$  be two unary relation symbols. An  $\{L, X\}$ -labelled tree is an  $\{E, L, X\}$ -structure  $\mathcal{T} = (T, E^T, L^T, X^T)$  where  $(T, E^T)$  is a tree.

**Theorem 4.** *Let  $\tau$  be a vocabulary that consists of a binary relation symbol  $E$  and two unary relation symbols  $L$  and  $X$ . For every  $h \geq 1$  there is a  $\text{FO}(\tau)$ -sentence  $\varphi_h$  of size  $2^{\mathcal{O}(h)}$  with the following properties:*

1.  $\varphi_h$  is preserved under extensions on the class of all  $\tau$ -structures, and
2. every existential  $\text{FO}(\tau)$ -sentence  $\psi$  that is equivalent to  $\varphi_h$  on the class  $\mathfrak{T}_{\leq h}$  of all  $\{L, X\}$ -labelled trees of height at most  $h$  is of size at least  $\text{Tower}(h-1)$ .

Using the same approach as in the previous sections, i.e., the encoding of natural numbers by trees introduced in Section 3, it is not difficult to construct a sentence  $\varphi_h$  of small size which meets requirement 2. We were, however, unable to find a sentence based on this encoding which also meets requirement 1 (even when considering  $\mathfrak{T}_{\leq h}$  instead of the class of all  $\tau$ -structures). To prove Theorem 4, we therefore introduce the following encoding of numbers by  $\{L, X\}$ -labelled trees. The remainder of this section is devoted to the proof of Theorem 4.

From now on, until the end of this section, we let  $\tau$  denote a vocabulary that consists of a binary relation symbol  $E$  and two unary relation symbols  $L$  and  $X$ .

**Definition 2.** For each natural number  $h \geq 1$  and each  $n \in \{0, 1, \dots, \text{Tower}(h)-1\}$ , we define the  $\{L, X\}$ -labelled tree  $\tilde{\mathcal{T}}_h(n)$  as follows:

- $\tilde{\mathcal{T}}_1(0)$  consists of two nodes  $u$  and  $v$  such that there is an edge from  $u$  to  $v$ , and  $u$  is labelled to be a *leaf* (which is encoded by “ $v \in L$ ”) and  $v$  is labelled  $\mathbf{0}$  (which is encoded by “ $v \notin X$ ”).
- $\tilde{\mathcal{T}}_1(1)$  consists of two nodes  $u$  and  $v$  such that there is an edge from  $u$  to  $v$ , and  $u$  is labelled to be a *leaf* (which is encoded by “ $v \in L$ ”) and  $v$  is labelled  $\mathbf{1}$  (which is encoded by “ $v \in X$ ”).
- for  $h \geq 1$  and  $n \in \{0, \dots, \text{Tower}(h+1)-1\} = \{0, \dots, 2^{\text{Tower}(h)}-1\}$ , the  $\{L, X\}$ -labelled tree  $\tilde{\mathcal{T}}_{h+1}(n)$  is obtained by creating a new root, attaching to it one copy of  $\tilde{\mathcal{T}}_h(i)$ , for each  $i \in \{0, \dots, \text{Tower}(h)-1\}$ , and labelling the root of  $\tilde{\mathcal{T}}_h(i)$  with  $\mathbf{1}$  if  $\text{bit}(i, n) = 1$ , and  $\mathbf{0}$  otherwise.

Note that for every fixed  $h$ , the trees  $\tilde{T}_h(n)$  for  $n < Tower(h)$  all have the same shape and only vary in the labelling (w.r.t.  $\mathbf{0}$  and  $\mathbf{1}$ ) of the children of the root. Furthermore, each path from the root of  $\tilde{T}_h(n)$  to a leaf has exactly length  $h$  (i.e., consists of  $h$  edges), and the nodes that are labelled  $L$  are exactly the *leaves* of  $\tilde{T}_h(n)$ .

Unlike in the previous sections, it does not suffice to restrict attention to structures that are obtained as disjoint unions or similar, easy combinations of the trees  $\tilde{T}_h(n)$ . Instead, we will consider a suitable notion where a node  $t$  in an arbitrary  $\tau$ -structure  $\mathcal{A}$  is called “ $h$ -good” if the substructure  $\mathcal{A}_t$  is “sufficiently similar” to the tree  $\tilde{T}_h(n)$ , for a number  $n < Tower(h)$ . The precise definition of this notion is given below. Before introducing it, however, we need the following (easy) lemma.

**Lemma 4.** *For every  $h' \geq 1$  there is a universal  $FO(\tau)$ -sentence  $forest_{\leq h'}$  of length  $\mathcal{O}(h')$  such that for every finite  $\tau$ -structure  $\mathcal{A} = (A, E^A, L^A, X^A)$  the following is true:  $\mathcal{A} \models forest_{\leq h'} \iff (A, E^A)$  is a disjoint union of trees such that every node in  $L^A$  is a leaf, and for every root  $r$  in  $\mathcal{A}$  (i.e., for every node  $r$  in  $A$  that has in-degree  $0$  in  $E^A$ ) the following is true: every path in  $\mathcal{A}$  that starts in  $r$  has length at most  $h'$ .*

**Definition 3** ( *$h$ -good nodes  $x$ , and the numbers  $Rep_h^A(x)$  represented by them*). Let  $h' \geq 1$  and let  $\mathcal{A}$  be a structure with  $\mathcal{A} \models forest_{\leq h'}$ . By induction on  $h \in \{1, \dots, h'\}$  we define the following notion:

A node  $x$  of  $\mathcal{A}$  is called *1-good* in  $\mathcal{A}$  iff it has at least one child  $y$  with  $L^A(y)$ , and for all children  $y'$  of  $x$  in  $\mathcal{A}$  the following is true: if  $L^A(y')$ , then  $X^A(y') \leftrightarrow X^A(y)$ .

Every 1-good node  $x$  in  $\mathcal{A}$  represents a number  $Rep_1^A(x) \in \{0, 1\}$  as follows:

$$Rep_1^A(x) = 0 \iff x \text{ has a child that belongs to } L^A \text{ but not to } X^A$$

$$Rep_1^A(x) = 1 \iff x \text{ has a child that belongs to } L^A \text{ and to } X^A.$$

Let  $h < h'$  be such that the notion of  $h$ -goodness as well as the numbers  $Rep_h^A(y)$ , for all  $h$ -good nodes  $y$  in  $\mathcal{A}$ , are already defined. Then, a node  $x$  of  $\mathcal{A}$  is called *( $h+1$ )-good* in  $\mathcal{A}$  iff the following is true: For each number  $i \in \{0, \dots, Tower(h)-1\}$  there exists a  $h$ -good child  $y_i$  of  $x$  in  $\mathcal{A}$  with  $Rep_h^A(y_i) = i$ , and for all  $h$ -good children  $z$  of  $x$  in  $\mathcal{A}$  with  $Rep_h^A(z) = i$  the following is true:  $X^A(z) \leftrightarrow X^A(y_i)$ .

Every  $(h+1)$ -good node  $x$  in  $\mathcal{A}$  represents the (uniquely defined) number

$$Rep_{h+1}^A(x) = n \in \{0, 1, \dots, 2^{Tower(h)}-1\} = \{0, 1, \dots, Tower(h+1)-1\}$$

which satisfies the following: for every  $i < Tower(h)$ ,  $bit(i, n) = 1 \iff X^A(y_i)$ .

The following notion of  *$h$ -inconsistency* can be viewed as a counterpart to the notion of  $h$ -goodness. Note, however, that  $h$ -goodness is a property of a *node* whereas  $h$ -inconsistency is a property of a whole structure.

**Definition 4** ( *$h$ -inconsistency*). Let  $h' \geq 1$  and let  $\mathcal{A}$  be a structure with  $\mathcal{A} \models forest_{\leq h'}$ . By induction on  $h \in \{1, \dots, h'\}$ , we define the following notion: We say that  $\mathcal{A}$  is *1-inconsistent* if there exist nodes  $x, y, y'$  such that  $y$  and  $y'$  are children of  $x$  with the following properties:  $y$  and  $y'$  both belong to  $L^A$ , and we have  $X^A(y)$  and  $\neg X^A(y')$ . Let  $h < h'$  be such that the notion of  $h$ -inconsistency is already defined.

We say that  $\mathcal{A}$  is  $(h+1)$ -inconsistent if there exist nodes  $x, y, y'$  such that  $y$  and  $y'$  are children of  $x$  with the following properties:  $y$  and  $y'$  both are  $h$ -good in  $\mathcal{A}$  with  $\text{Rep}_h^{\mathcal{A}}(y) = \text{Rep}_h^{\mathcal{A}}(y')$ , and we have  $X^{\mathcal{A}}(y)$  and  $\neg X^{\mathcal{A}}(y')$ .

Furthermore, we say that  $\mathcal{A}$  is  $(\leq h)$ -inconsistent if there exists a  $\tilde{h} \in \{1, \dots, h\}$  such that  $\mathcal{A}$  is  $\tilde{h}$ -inconsistent. It is straightforward (but tedious) to show the following:

**Lemma 5.** *For every  $h \geq 1$  there is a  $\text{FO}(\tau)$ -sentence  $\varphi_h$  of size  $2^{\mathcal{O}(h)}$  such that the following is true for every  $\tau$ -structure  $\mathcal{A}$ :  $\mathcal{A} \models \varphi_h \iff \mathcal{A} \models \neg \text{forest}_{\leq h}$  or  $\mathcal{A}$  is  $(\leq h)$ -inconsistent or there exists a node  $x$  that is  $h$ -good in  $\mathcal{A}$ .*

Furthermore, it can be shown that this sentence  $\varphi_h$  is preserved under extensions. This finally enables us to prove Theorem 4.

## 7 Structures of Bounded Degree — Elementary Upper Bounds

All the non-elementary lower bounds in previous sections depend heavily on the fact that we consider classes of structures of unbounded degree. On classes of structures of bounded degree, the picture looks entirely different as we can prove elementary upper bounds as counterparts to Theorems 1, 3, and 4. Throughout the remainder of this section we let  $\tau$  be a fixed finite relational vocabulary, and we let  $d$  be a fixed natural number. We write  $\mathfrak{D}_d$  to denote the class of all  $\tau$ -structures whose Gaifman graph has degree at most  $d$ . By an easy adaptation of the model theoretic proof of Gaifman’s theorem given in [5], one obtains the following elementary upper bound:

**Theorem 5.** *There is a 4-fold exponential function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that for every  $\text{FO}(\tau)$ -sentence  $\varphi$  there is a sentence  $\psi$  of size  $\leq g(\|\varphi\|)$  with the following properties:  $\psi$  is a Boolean combination of basic local sentences and  $\psi$  is equivalent to  $\varphi$  on all structures in  $\mathfrak{D}_d$ .*

By similar techniques we can prove an elementary upper bound for the Feferman-Vaught theorem stratified by formula length. Furthermore, there are elementary decision algorithms for the first-order theories of classes of trees of bounded arity, in particular for the class of binary trees. Refining the methods of [11], one also obtains an elementary upper bound for the following variant of the Łoś-Tarski Theorem.

**Theorem 6.** *There is a 5-fold exponential function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that any  $\text{FO}(\tau)$ -sentence  $\varphi$  that is preserved under extensions on the class of acyclic structures in  $\mathfrak{D}_d$  is equivalent, on this class, to an existential first-order sentence of length at most  $f(\|\varphi\|)$ .*

In all the above cases for structures of bounded degree we can also prove at least 2-fold exponential lower bounds.

**Acknowledgements.** We would like to thank an anonymous referee for pointing us to the references [2, 7]. We gratefully acknowledge the support of the Isaac Newton Institute through its 2006 programme on Logic and Algorithms. The opportunity afforded by this programme greatly aided our collaboration.

## References

1. Atserias, A., Dawar, A., Grohe, M.: Preservation under extensions on well-behaved finite structures. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1437–1449. Springer, Heidelberg (2005)
2. Compton, K., Henson, C.W.: A uniform method for proving lower bounds on the computational complexity of logical theories. In: Abramsky, S., Gabbay, D.M., Maibaum, T. (eds.) Handbook of Logic in Computer Science. Logic and Algebraic Methods, vol. 5, pp. 129–216. Oxford University Press, Oxford (2000)
3. Cutland, N.J.: Computability. Cambridge University Press, Cambridge (1980)
4. Dawar, A., Grohe, M., Kreutzer, S., Schweikardt, N.: Approximation schemes for first-order definable optimisation problems. In: Proc. LICS'06, pp. 411–420 (2006)
5. Ebbinghaus, H.-D., Flum, J.: Finite Model Theory, 2nd edn. Springer, Heidelberg (1999)
6. Feferman, S., Vaught, R.: The first order properties of products of algebraic systems. *Fundamenta Mathematicae* 47, 57–103 (1959)
7. Ferrante, J., Rackoff, C.: Computational Complexity of Logical Theories. Lecture Notes in Mathematics, vol. 718. Springer, Heidelberg (1979)
8. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006)
9. Frick, M., Grohe, M.: Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM* 48, 1184–1206 (2001)
10. Frick, M., Grohe, M.: The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic* 130, 3–31 (2004)
11. Gaifman, H.: On local and non-local properties. In: Stern, J. (ed.) Proceedings of the Herbrand Symposium, Logic Colloquium '81, pp. 105–135. North-Holland, Amsterdam (1982)
12. Grohe, M., Schweikardt, N.: Comparing the succinctness of monadic query languages over finite trees. *RAIRO: Theoretical Informatics and Applications (ITA)* 38, 343–373 (2005)
13. Grohe, M., Schweikardt, N.: The succinctness of first-order logic on linear orders. *Logical Methods in Computer Science* 1(1:6), 1–25 (2005)
14. Gurevich, Y.: Toward logic tailored for computational complexity. In: Richter, M., et al. (eds.) Computation and Proof Theory. Lecture Notes in Mathematics, vol. 1104, pp. 175–216. Springer, Heidelberg (1984)
15. Hodges, W.: Model Theory. Cambridge University Press, Cambridge (1993)
16. Libkin, L.: On forms of locality over finite models. In: Proceedings of the 12th IEEE Symposium on Logic in Computer Science, pp. 204–215. IEEE Computer Society Press, Los Alamitos (1997)
17. Libkin, L.: Logics with counting and local properties. *Transaction on Computational Logic* 1, 33–59 (2000)
18. Pan, G., Vardi, M.: Fixed-parameter hierarchies inside PSPACE. In: Proceedings of the 21st IEEE Symposium on Logic in Computer Science, pp. 27–36. IEEE Computer Society Press, Los Alamitos (2006)
19. Rabin, M.: Decidability of second order theories and automata on infinite trees. *Transactions of the American Mathematical Society* 141, 1–35 (1969)
20. Rossman, B.: Existential positive types and preservation under homomorphisms. In: 20th IEEE Symposium on Logic in Computer Science, pp. 467–476. IEEE Computer Society Press, Los Alamitos (2005)
21. Stockmeyer, L., Meyer, A.: Word problems requiring exponential time. In: Proceedings of the 5th ACM Symposium on Theory of Computing, pp. 1–9. ACM Press, New York (1973)
22. Tait, W.W.: A counterexample to a conjecture of Scott and Suppes. *Journal of Symbolic Logic* 24, 15–16 (1959)
23. Thatcher, J., Wright, J.: Generalised finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory* 2, 57–81 (1968)

# Decision Problems for Lower/Upper Bound Parametric Timed Automata<sup>\*</sup>

Laura Bozzelli<sup>1</sup> and Salvatore La Torre<sup>2</sup>

<sup>1</sup> Università di Napoli Federico II , Via Cintia, 80126 - Napoli, Italy

<sup>2</sup> Università degli Studi di Salerno, Via Ponte Don Melillo - 84084 Fisciano, Italy

**Abstract.** We investigate a class of parametric timed automata, called lower bound/upper bound (L/U) automata, where each parameter occurs in the timing constraints either as a lower bound or as an upper bound. For such automata, we show that checking if for a parameter valuation (resp., all parameter valuations) there is an infinite accepting run is PSPACE-complete. We extend these results by allowing the specification of constraints on parameters as a linear system. We show that the considered decision problems are still PSPACE-complete, if the lower bound parameters are not compared to the upper bound parameters in the linear system, and are undecidable in general. Finally, we consider a parametric extension of MITL<sub>0,∞</sub>, and prove that the related satisfiability and model checking (w.r.t. L/U automata) problems are PSPACE-complete.

## 1 Introduction

Timed automata [2] are a widely accepted formalism to model the behavior of real-time systems. A timed automaton is a finite-state transition graph equipped with a finite set of *clock variables* which are used to express *timing constraints*. The semantics is given by an infinite-state transition system where transitions correspond either to a change of location (instantaneous transition) or to a time consumption (time transition). Over the years, timed automata have been intensively studied by many authors, and significant progresses have been done in developing verification algorithms, heuristics, and tools (see [6] for a recent survey).

Timing constraints in timed automata allow the specification of constant bounds on delays among events. Typical examples are upper and lower bounds on computation times, message delays and timeouts. In the early stages of a design, when not much is known about the system under development, it is however useful for designers to use parameters instead of specific constants.

In [5], Alur et al. introduce parametric timed automata, i.e., timed automata where clocks can be compared to parameters. For such class of automata, they study the *emptiness problem*: “is there a parameter valuation for which the automaton has an accepting run?” This problem turns out to be undecidable already for parametric timed automata with only three parametric clocks, while it is decidable when at most one clock is compared to parameters. In case of two parametric clocks, the emptiness problem is closely

---

<sup>\*</sup> This research was partially supported by the MIUR grant ex-60% 2005-2006 Università di Salerno, and the European Commission via FP6 program under contracts FP6-1596 AEOLUS.

related to various hard and open problems of logic and automata theory [5]. In [11], Hune et al. identify a subclass of parametric timed automata, called *lower bound/upper bound (L/U) automata*, in which each parameter occurs either as a lower bound or as an upper bound in the timing constraints. Despite this limitation, the model is still interesting in practice. In fact, L/U automata can be used to model the Fisher’s mutual exclusion algorithm [13], the root contention protocol [12] and other known examples from the literature (see [11]). Hune et al. show that the emptiness problem for L/U automata with respect to finite runs is decidable. The case of *infinite* accepting runs (which is crucial for the verification of liveness properties) is not investigated, and does not follow from their results.

In this paper, we further investigate the class of L/U automata and consider acceptance conditions over infinite runs. Given an L/U automaton  $\mathcal{A}$ , denote with  $\Gamma(\mathcal{A})$  the set of parameter valuations for which the automaton has an infinite accepting run. We show that questions about  $\Gamma(\mathcal{A})$  can be answered considering a bounded set of parameter valuations of size exponential in the size of the constants and the number of clocks, and polynomial in the number of parameters and locations of  $\mathcal{A}$ . Therefore, we are able to show that checking the set  $\Gamma(\mathcal{A})$  for emptiness and universality (i.e., if  $\Gamma(\mathcal{A})$  contains all the parameter valuations) is PSPACE-complete. The main argument for such results is as follows: suppose that  $\mathcal{A}$  is an L/U automaton which uses parameters only as either upper bounds or lower bounds; then if an infinite run  $\rho$  is accepted by  $\mathcal{A}$  for large-enough values of the parameters, we can determine appropriate finite portions of  $\rho$  which can be “repeatedly simulated” (resp., “deleted”) thus obtaining a run  $\rho'$  which is accepted by  $\mathcal{A}$  for larger (resp., smaller) parameters values.

Parameters in system models can be naturally related by linear equations and inequalities. As an extension of the above results, we consider *constrained emptiness* and *constrained universality* on L/U automata, where the constraint is represented by a linear system over parameters. We show that these problems are in general undecidable, and become decidable in polynomial space (and thus PSPACE-complete) if we do not compare parameters of different types in the linear constraint.

An important consequence of our results on L/U automata is the extension to the dense-time paradigm of the results shown in [3]. We define a parametric extension of the temporal logic  $\text{MITL}_{0,\infty}$  [4], denoted  $\text{PMITL}_{0,\infty}$ , and show that (under restrictions on the use of parameters analogous to those imposed on L/U automata) the related satisfiability and model-checking problems are PSPACE-complete. The proof consists of translating formulas to L/U automata. To the best of our knowledge this is the first work that solves verification problems against linear-time specifications with parameters both in the model and in the specification.

Besides the already mentioned research, there are several other papers that are related to ours. The idea of restricting the use of parameters (in order to obtain decidability) such that upper and lower bounds cannot share a same parameter is also present in [3] where the authors study the logic LTL [14] augmented with parameters. The general structure of our argument for showing decidability (“pumping” argument) is inspired to their approach. However, let us stress that there are substantial technical differences with that paper since we consider a different framework, and in particular, we deal with a dense-time semantics. Parametric branching time specifications were first

investigated in [16,9] where decidability is shown for logics obtained as extensions of TCTL [1] with parameters. In [7], decidability is extended to full TCTL with Presburger constraints over parameters. In [8], decidability is established for the model checking problem of *discrete-time* timed automata with *one parametric clock* against parametric TCTL without equality (for full TCTL with parameters the problem is undecidable). Finally, recall that the undecidability of systems with parameters is also captured by the undecidability results shown in [10]. However, the limitations we consider for obtaining decidability seem to be orthogonal to those considered there. We are not aware of any way of obtaining our decidability results from those presented in [10].

Due to the lack of space, for the omitted details we refer the interested reader to a forthcoming extended version of this paper.

## 2 Parametric Timed Automata

Throughout this paper, we fix a finite set of *parameters*  $P = \{p_1, \dots, p_m\}$ . Let  $\mathbb{R}_{\geq 0}$  be the set of non-negative reals,  $\mathbb{N}$  the set of natural numbers, and  $\mathbb{Z}$  the set of integers.

A *linear expression*  $e$  is an expression of the form  $c_0 + c_1p_1 + \dots + c_m p_m$  with  $c_0, c_1, \dots, c_m \in \mathbb{Z}$ . We say that parameter  $p_i$  *occurs* in  $e$  if  $c_i \neq 0$ . A (*parameter*) *valuation* is a function  $v : P \rightarrow \mathbb{N}$  assigning a natural number to each parameter. The *null parameter valuation*, denoted  $v_{null}$ , is the valuation assigning 0 to each parameter. For the linear expression  $e$  above,  $e[v]$  denotes the integer  $c_0 + c_1v(p_1) + \dots + c_mv(p_m)$ .

We fix a finite set of *clocks*  $X$ . For the ease of presentation, we allow in our model a special clock  $x_0 \in X$ , called *zero clock*, which always evaluates to 0 (i.e., it does not increase with time).

An *atomic (clock) constraint*  $f$  is an expression of the form  $x - y \prec e$ , where  $x, y \in X$ ,  $e$  is a linear expression, and  $\prec \in \{<, \leq\}$ . We say that  $f$  is *parametric* if some parameter occurs in  $e$ . A (*clock*) *constraint* is a finite conjunction of atomic constraints. A *clock valuation* is a function  $w : X \rightarrow \mathbb{R}_{\geq 0}$  assigning a value in  $\mathbb{R}_{\geq 0}$  to each clock and s.t.  $w(x_0) = 0$ . For a constraint  $f$ , a parameter valuation  $v$ , and a clock valuation  $w$ , the pair  $(v, w)$  *satisfies*  $f$ , denoted  $(v, w) \models f$ , if the expression obtained from  $f$  by replacing each parameter  $p$  with  $v(p)$  and each clock  $x$  with  $w(x)$  evaluates to true.

A *reset set*  $r$  is a subset of  $X$  containing the clocks to be reset to 0. For  $\tau \in \mathbb{R}_{\geq 0}$  and a clock valuation  $w$ , the clock valuation  $w + \tau$  is defined as  $(w + \tau)(x) = w(x) + \tau$  for all  $x \in X \setminus \{x_0\}$  and  $(w + \tau)(x_0) = 0$ . For a reset set  $r \in 2^X$ , the clock valuation  $w[r]$  is defined as  $w[r](x) = 0$  if  $x \in r$  and  $w[r](x) = w(x)$  otherwise. Let  $\Xi$  be the set of all clock constraints over  $X$  and  $P$ .

**Definition 1.** A parametric timed automaton (PTA) is a tuple  $\mathcal{A} = \langle Q, q^0, \Delta, F \rangle$ , where  $Q$  is a finite set of locations,  $q^0 \in Q$  is the initial location,  $\Delta \subseteq Q \times \Xi \times 2^X \times Q$  is a finite transition relation, and  $F \subseteq Q$  is a set of accepting locations.

Let  $\mathcal{A} = \langle Q, q^0, \Delta, F \rangle$  be a PTA. A *state* of  $\mathcal{A}$  is a pair  $(q, w)$  such that  $q \in Q$  is a location and  $w$  is a clock valuation. The *initial state* is  $(q^0, \vec{0})$ , where  $\vec{0}$  maps every  $x \in X$  to 0. We denote by  $X(P)$  the set of *parametric clocks*, that is the set of  $x \in X$  such that  $\mathcal{A}$  contains a parametric atomic constraint of the form  $x - y \prec e$  or  $y - x \prec e$ . A PTA  $\mathcal{A}$  is called a *timed automaton* (TA, for short), if  $\mathcal{A}$  does not contain occurrences



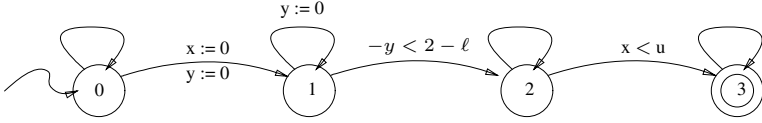


Fig. 1. An L/U automaton

of parameters. For a PTA  $\mathcal{A}$  and a parameter valuation  $v$ , we denote by  $\mathcal{A}_v$  the TA obtained by replacing each linear expression  $e$  of  $\mathcal{A}$  by  $e[v]$ .

Let  $\mathcal{A} = \langle Q, q^0, \Delta, F \rangle$  be a PTA and  $v$  be a parameter valuation. The concrete semantics of  $\mathcal{A}$  under  $v$ , denoted  $\llbracket \mathcal{A} \rrbracket_v$ , is the labelled transition system  $\langle S, \Rightarrow \rangle$  over  $(\Delta \cup \{\perp\}) \times \mathbb{R}_{\geq 0}$ , where  $S$  is the set of  $\mathcal{A}$  states and for  $\tau \geq 0$ ,  $(q, w) \xrightarrow{\delta, \tau} (q', w')$  iff:

- either  $\delta = (q, g, r, q')$ ,  $\tau = 0$ ,  $(v, w) \models g$ , and  $w' = w[r]$  (instantaneous transition),
- or  $\delta = \perp$ ,  $q' = q$ , and  $w' = w + \tau$  (time transition).

An infinite run of  $\llbracket \mathcal{A} \rrbracket_v$  is an infinite path  $\rho = s_0 \xrightarrow{\delta_0, \tau_0} s_1 \xrightarrow{\delta_1, \tau_1} s_2 \dots$  of  $\llbracket \mathcal{A} \rrbracket_v$  such that  $\sum_{i \geq 0} \tau_i = \infty$  (progress condition) and for infinitely many  $i \geq 0$ ,  $\delta_i \neq \perp$  (there are infinitely many occurrences of instantaneous transitions). Moreover,  $\rho$  is accepting iff for infinitely many  $i \geq 0$ , we have that  $q_i \in F$ , where  $s_i = (q_i, w_i)$ . A finite run of  $\llbracket \mathcal{A} \rrbracket_v$  is a finite path  $\rho = s_0 \xrightarrow{\delta_0, \tau_0} s_1 \dots s_{n-1} \xrightarrow{\delta_{n-1}, \tau_{n-1}} s_n$  of  $\llbracket \mathcal{A} \rrbracket_v$ . The duration of  $\rho$ , denoted by  $DUR(\rho)$ , is defined as  $DUR(\rho) = \sum_{i=0}^{n-1} \tau_i$ . We denote with  $\Gamma(\mathcal{A})$  the set of parameter valuations  $v$  such that there exists an accepting infinite run of  $\llbracket \mathcal{A} \rrbracket_v$  from the initial state  $(q^0, \vec{0})$ .

Given a linear expression  $e = c_0 + c_1 p_1 + \dots + c_m p_m$  and a parameter  $p_i \in P$ , we say that  $p_i$  occurs positively in  $e$  if  $c_i \geq 0$ . Analogously, we say that  $p_i$  occurs negatively in  $e$  if  $c_i \leq 0$ . A lower bound parameter (resp., an upper bound parameter) of a PTA  $\mathcal{A}$  is a parameter that only occurs negatively (resp., occurs positively) in the expressions of  $\mathcal{A}$ . We call  $\mathcal{A}$  a lower bound/upper bound (L/U) automaton if every parameter occurring in  $\mathcal{A}$  is either an upper bound parameter or a lower bound parameter. Moreover, we say that  $\mathcal{A}$  is a lower bound automaton (resp., upper bound automaton) iff every parameter occurring in  $\mathcal{A}$  is a lower bound parameter (resp., an upper bound parameter).

**Example 1.** Consider the automaton  $\mathcal{A}$  in Fig. 1. It has four locations 0, 1, 2, 3, two clocks  $x, y$ , and two parameters  $\ell$  and  $u$ . Note that the constraint  $-y < 2 - \ell$  imposes a lower bound on the possible values of  $y$ , while  $x < u$  imposes an upper bound on the possible values of  $x$ . Thus,  $\ell$  and  $u$  are respectively a lower bound and an upper bound parameter, and  $\mathcal{A}$  is an L/U automaton. Also, it is easy to verify that  $\llbracket \mathcal{A} \rrbracket_v$  has an infinite run from location 0 visiting infinitely often location 3 iff  $v(\ell) < v(u) + 2$  and  $v(u) > 0$ . Therefore,  $\Gamma(\mathcal{A}) = \{v \mid v(\ell) < v(u) + 2 \text{ and } v(u) > 0\}$ .

For an L/U automaton  $\mathcal{A}$ , we consider the following decision problems on  $\Gamma(\mathcal{A})$ :

- *Emptiness*: is the set  $\Gamma(\mathcal{A})$  empty?
- *Universality*: does the set  $\Gamma(\mathcal{A})$  contain all parameter valuations?

**Relations over states.** For  $t \in \mathbb{R}_{\geq 0}$ ,  $\lfloor t \rfloor$  denotes the integral part of  $t$  and  $\text{fract}(t)$  denotes its fractional part. We define the following equivalence relations over  $\mathbb{R}_{\geq 0}$ :

- $t \approx t'$  iff (i)  $\lfloor t \rfloor = \lfloor t' \rfloor$  and (ii)  $\text{fract}(t) = 0$  iff  $\text{fract}(t') = 0$ ;
- for every  $K \in \mathbb{N}$ ,  $t \approx_K t'$  iff either  $t \approx t'$  or  $t, t' > K$ .

Let  $\mathcal{A} = \langle Q, q^0, \Delta, F \rangle$  be a PTA and  $v$  be a parameter valuation. We denote by  $K_v$  the largest  $|e[v]| + 1$  such that  $e$  is a linear expression of  $\mathcal{A}$ . The *region equivalence* of  $\mathcal{A}$  with respect to  $v$ , denoted  $\approx_v$ , is the equivalence relation over  $\mathcal{A}$  states defined as:  $(q, w) \approx_v (q', w')$  iff  $q = q'$  and for all clocks  $x, y \in X$ , (i)  $w(x) - w(y) \geq 0$  iff  $w'(x) - w'(y) \geq 0$ , (ii)  $|w(x) - w(y)| \approx_{K_v} |w'(x) - w'(y)|$ , (iii)  $\text{fract}(w(x)) \leq \text{fract}(w(y))$  iff  $\text{fract}(w'(x)) \leq \text{fract}(w'(y))$  (*ordering of fractional parts*).

A *region* of  $\mathcal{A}$  with respect to  $v$  is an equivalence class induced by  $\approx_v$ . Recall that the number of these regions is  $O(|Q| \cdot (2K_v + 2)^{|X|^2})$  [2] (note that we consider also diagonal constraints). Moreover,  $\approx_v$  is a bisimulation over  $\llbracket \mathcal{A} \rrbracket_v$ . Note that if  $\mathcal{A}$  is a timed automaton, then the value of  $K_v$  is obviously independent on specific valuation  $v$ , and we denote it with  $K_{\mathcal{A}}$ . Thus, the emptiness for a timed automaton is reduced to check emptiness of the finite-state quotient graph induced by region equivalence (*region graph*) [2].

**Theorem 1.** *Checking emptiness for a timed automaton  $\mathcal{A}$  is PSPACE-complete and can be done in time  $O(|\Delta| \cdot (2K_{\mathcal{A}} + 2)^{2|X|^2})$ .*

To answer questions on  $\Gamma(\mathcal{A})$ , for a parametric timed automaton  $\mathcal{A}$ , we need to examine an infinite class of region graphs, one for each parameter valuation. However, in the next sections we will show that for an L/U automaton  $\mathcal{A}$ , it is possible to effectively determine a parameter valuation  $v$  such that our decision problems can be reduced to check emptiness of  $\mathcal{A}_v$ . In our arguments, we use a preorder  $\sqsubseteq$  over the set of states defined as  $(q, w) \sqsubseteq (q', w')$  iff

- $(q, w) \approx_{v_{null}} (q', w')$  (recall that  $v_{null}$  is the null parameter valuation);
- for all clocks  $x, y \in X(P)$  such that  $w(x) - w(y) > 0$ : either  $w'(x) - w'(y) \geq w(x) - w(y)$ , or  $(w'(x) - w'(y)) \approx (w(x) - w(y))$  hold.

The first condition establishes that  $(q, w)$  and  $(q', w')$  are equivalent w.r.t. all non-parametric clock constraints. The second condition ensures that, for a lower (resp. upper) bound automaton, each parametric clock constraint which is fulfilled in  $(q, w)$  (resp.  $(q', w')$ ) is also fulfilled in  $(q', w')$  (resp.  $(q, w)$ ). We will show that  $\sqsubseteq$  indeed defines a simulation relation over the states of a lower (resp. upper) bound automaton.

For an L/U automaton  $\mathcal{A}$ , we will use the following constants:

- $k_{\mathcal{A}}$  denotes the number of parametric clocks of  $\mathcal{A}$ , i.e. the size of  $X(P)$ ;
- $c_{\mathcal{A}}$  is the maximum over  $\{|c| + 1 \mid \text{there is a linear expression of } \mathcal{A} \text{ of the form } c_0 + c_1p_1 + \dots + c_m p_m \text{ and } c = c_i \text{ for some } 0 \leq i \leq m\}$ .
- $N_{\mathcal{R}(\mathcal{A})}$  is the number of regions of  $\mathcal{A}$  with respect to the null parameter valuation.

### 3 Emptiness and Universality for Lower Bound Automata

In this section, we study the considered decision problems for lower bound automata. We fix a lower bound automaton  $\mathcal{A} = \langle Q, q^0, \Delta, F \rangle$ . Also, for two parameter valuations  $v_1$  and  $v_2$ , we write  $v_1 \leq v_2$  to mean that  $v_1(p) \leq v_2(p)$  for all  $p \in P$ .

**Emptiness.** We recall that every linear expression of  $\mathcal{A}$  is of the form  $c_0 - c_1p_1 - \dots - c_m p_m$  with  $c_i \in \mathbb{N}$  for  $1 \leq i \leq m$ . By decreasing the parameter values, the constraints of  $\mathcal{A}$  are weakened. Thus, if  $v \leq v'$  and  $v' \in \Gamma(\mathcal{A})$ , then also  $v \in \Gamma(\mathcal{A})$  (i.e.,  $\Gamma(\mathcal{A})$  is downward-closed). Hence, to test emptiness of  $\Gamma(\mathcal{A})$  it suffices to check emptiness of the TA  $\mathcal{A}_{v_{null}}$ . By Theorem 1 we obtain:

**Theorem 2.** *Given a lower bound automaton  $\mathcal{A}$ , checking emptiness of  $\Gamma(\mathcal{A})$  is PSPACE-complete and can be done in time  $O(|\Delta| \cdot (2c_{\mathcal{A}} + 2)^{2|X|^2})$ .*

**Universality.** For checking universality of  $\Gamma(\mathcal{A})$ , we define a parameter valuation  $v_{\mathcal{A}}$  (assigning “large” values to parameters) and show that if  $v_{\mathcal{A}} \in \Gamma(\mathcal{A})$  then each  $v \geq v_{\mathcal{A}}$  also belongs to  $\Gamma(\mathcal{A})$ . Since  $\Gamma(\mathcal{A})$  is downward closed, checking universality of  $\Gamma(\mathcal{A})$  reduces to checking if  $v_{\mathcal{A}} \in \Gamma(\mathcal{A})$ , and thus, checking for non-emptiness of the timed automaton  $\mathcal{A}_{v_{\mathcal{A}}}$ .

Define  $N_{\mathcal{A}}$  as the constant  $k_{\mathcal{A}}(N_{\mathcal{R}(\mathcal{A})} + 1) + c_{\mathcal{A}}$ , and denote by  $v_{\mathcal{A}}$  the parameter valuation assigning  $N_{\mathcal{A}}$  to each parameter. The choice of such a large constant is to ensure that in any run  $\rho$  of  $\llbracket \mathcal{A} \rrbracket_{v_{\mathcal{A}}}$  we can find subruns  $\rho'$  that can be repeatedly and consecutively simulated such that we can construct a corresponding run for  $\llbracket \mathcal{A} \rrbracket_v$ , for any  $v \geq v_{\mathcal{A}}$ . Intuitively,  $N_{\mathcal{A}}$  is sufficiently large to ensure that there is a portion  $\rho'$  of  $\rho$  (of duration larger than 1) which corresponds to a cycle of  $\mathcal{A}_{v_{null}}$  and such that each parametric clock constraints is either always or never satisfied in all the states visited along  $\rho'$ .

A parameter valuation  $v$  evaluates negative for  $\mathcal{A}$  if for each parametric atomic constraint  $x - y \prec e$  of  $\mathcal{A}$ ,  $e[v] < 0$ . Note that  $v_{\mathcal{A}}$  evaluates negative for  $\mathcal{A}$ . We give two technical lemmas that will be used in the proof of the main theorem of this section. In these two lemmas,  $v$  is a parameter valuation which evaluates negative for  $\mathcal{A}$ .

**Lemma 1.** [Simulation Lemma for Lower Bound Automata] *Let  $\rho = s_0 \xrightarrow{\delta_0, \tau_0} s_1 \xrightarrow{\delta_1, \tau_1} \dots$  be a run of  $\llbracket \mathcal{A} \rrbracket_v$  and  $s'_0 \sqsupseteq s_0$ . Then, there is a run of  $\llbracket \mathcal{A} \rrbracket_v$  of the form  $\rho' = s'_0 \xrightarrow{\delta_0, \tau'_0} s'_1 \xrightarrow{\delta_1, \tau'_1} \dots$  such that  $s'_i \sqsupseteq s_i$  for each  $i$ , and  $DUR(\rho') \approx DUR(\rho)$  if  $\rho$  is finite.*

The following lemma allows us to append to a run in  $\llbracket \mathcal{A} \rrbracket_v$ , which corresponds to a cycle in the region graph of  $\mathcal{A}_{v_{null}}$ , another cycle such that its initial state  $s$  and its final state  $s'$  satisfy the strongest condition  $s \sqsubseteq s'$ . Note that once we apply this lemma, further cycles can be appended by repeatedly applying the Simulation Lemma. Also, note that from classical properties of timed automata the Simulation Lemma continues to hold if we replace  $\sqsubseteq$  with the region equivalence  $\approx_v$ . However, this does not hold for the following Lemma (the properties of  $\sqsubseteq$  are crucial).

**Lemma 2.** *Let  $\rho = s_0 \xrightarrow{\delta_0, \tau_0} s_1 \dots s_{n-1} \xrightarrow{\delta_{n-1}, \tau_{n-1}} s_n$  be a run of  $\llbracket \mathcal{A} \rrbracket_v$  such that  $s_0 \approx_{v_{null}} s_n$  and for every parametric clock  $x \in X(P) \setminus \{x_0\}$ , if a parametric atomic constraint of the form  $y - x \prec e$  appears along  $\rho$  then  $x$  is never reset along  $\rho$ . Then, there is a run  $\rho' = s'_0 \xrightarrow{\delta_0, \tau'_0} s'_1 \dots s'_{n-1} \xrightarrow{\delta_{n-1}, \tau'_{n-1}} s'_n$  of  $\llbracket \mathcal{A} \rrbracket_v$  such that  $DUR(\rho') \approx DUR(\rho)$ ,  $s'_0 = s_n$ , and  $s'_0 \sqsubseteq s'_n$ .*

In the next theorem we show that  $v_{\mathcal{A}}$  is the key valuation for reducing universality to membership to  $\Gamma(\mathcal{A})$ .

**Theorem 3.** *Let  $v, v'$  be parameter valuations such that  $v' \geq v \geq v_{\mathcal{A}}$ . Then,  $v \in \Gamma(\mathcal{A})$  implies  $v' \in \Gamma(\mathcal{A})$ .*

**Proof of Theorem 3:** Let  $v, v'$  be parameter valuations such that  $v' \geq v \geq v_{\mathcal{A}}$ . We can assume that each parameter appears precisely once in  $\mathcal{A}$ . In fact, if a parameter  $p$  appears twice, we can rename the second occurrence to  $p'$  and let  $v(p') = v(p)$  and  $v'(p') = v'(p)$ . Note that this assumption does not affect the constant  $N_{\mathcal{A}}$  which depends on the number of parameterized clocks and not on the number of parameters.

Fix a parameter  $p$  of  $\mathcal{A}$ . Let  $f_p = z - y \prec e$  be the unique atomic constraint of  $\mathcal{A}$  such that  $p$  occurs in  $e$ . We define  $v_p$  such that  $v_p$  assigns the value  $v(p) + 1$  to  $p$  and  $v(p')$  to all the other parameters  $p'$ . Since we can obtain  $v'$  from  $v$  by a sequence of steps, where a step corresponds to incrementing only one parameter by 1, it suffices to prove:

$$v \in \Gamma(\mathcal{A}) \text{ implies } v_p \in \Gamma(\mathcal{A}) \quad (1)$$

Observe that since  $v \geq v_{\mathcal{A}}$  and  $\mathcal{A}$  is a lower bound automaton, we have that  $v$  evaluates negative for  $\mathcal{A}$ , and in particular,  $e[v] < 0$ . Therefore, if  $y$  is the zero clock  $x_0$ ,  $f_p$  is unsatisfiable under valuation  $v$  and Assertion (II) trivially holds. Consider now the case  $y \neq x_0$  and also assume that  $z \neq x_0$  (the other case being simpler).

Let  $\rho = s_0 \xrightarrow{\delta_0, \tau_0} s_1 \xrightarrow{\delta_1, \tau_1} s_2 \dots$  be an infinite accepting run of  $\llbracket \mathcal{A} \rrbracket_v$  where  $s_i = (q_i, w_i)$  for  $i \geq 0$  and such that clock  $y$  is zero in  $s_0$  (note that if  $s_0$  is the initial state of  $\mathcal{A}$ , this last condition is satisfied). Then, we need to show that there is an infinite accepting run  $\rho'$  in  $\llbracket \mathcal{A} \rrbracket_{v_p}$  from  $s_0$ . In the following, for  $i \leq j$ , denote

$$\rho[i, j] = s_i \xrightarrow{\delta_i, \tau_i} \dots \xrightarrow{\delta_{j-1}, \tau_{j-1}} s_j.$$

In the rest of the proof, we first determine a finite portion of the run  $\rho$  that is crucial for the satisfaction of  $f_p$  under valuation  $v_p$  and suitable for repeated simulation, i.e., such that it meets the hypothesis of Lemma 2. Then, simulate this finite run an arbitrary number of times by applying Lemma 2 for the first simulation and Lemma 1 for the remaining ones. We end with the simulation of the remaining suffix of the run  $\rho$  applying again Lemma 1. The process is iterated until the resulting run is a run of  $\llbracket \mathcal{A} \rrbracket_{v_p}$ .

Assume that the clock constraint  $f_p$  appears along  $\rho$  (in the other case,  $\rho$  is also a run of  $\llbracket \mathcal{A} \rrbracket_{v_p}$ ), and let  $M$  be the smallest index such that  $f_p$  is in the clock constraint of transition  $\rho[M, M + 1]$ . Thus,  $(v, w_M) \models f_p$ . Since  $f_p = z - y \prec e$  and  $e[v] \leq e[v_{\mathcal{A}}] < c_{\mathcal{A}} - N_{\mathcal{A}}$ , by simple arguments, it is possible to show that there are  $M_y, M_z \in [0, M]$  such that:  $M_y < M_z$ ,  $w_{M_y}(y) = 0$ ,  $w_{M_z}(z) = 0$ , clock  $y$  is never reset along  $\rho[M_y, M_z]$ , and  $DUR(\rho[M_y, M_z]) > N_{\mathcal{A}} - c_{\mathcal{A}}$ .

Observe that in a run, each time transition can be split into an arbitrary number of time transitions. Thus, we can assume without loss of generality that for every  $\tau \in \mathbb{N}$ , there is  $i \geq M_y$  such that  $DUR(\rho[M_y, i]) = \tau$ . The following claim allows us to apply Lemma 2. Its proof relies on a counting argument that uses the constant  $N_{\mathcal{A}}$ , and thus also gives a more concrete explanation of our choice for its value.

*Claim.* There is an interval  $[i, j] \subseteq [M_y, M_z]$  such that  $DUR(\rho[i, j]) \geq 1$ ,  $s_i \approx_{v_{null}} s_j$ , and for every clock  $x \in X(P) \setminus \{x_0\}$ : if a parametric atomic constraint of the form  $x' - x \prec e'$  appears along  $\rho[i, j]$ , then  $x$  is never reset along  $\rho[i, j]$ .

*Proof of the Claim:* Let  $M_y \leq K \leq M_z$  be such that  $DUR(\rho[M_y, K]) = N_A - c_A$  (recall that  $DUR(\rho[M_y, M_z]) > N_A - c_A$ ). Let  $Y = \{x_1, \dots, x_n\}$  with  $n \leq k_A - 1$  be the set of clocks in  $X(P) \setminus \{x_0\}$  which are reset along  $\rho[M_y, K]$  and for  $h = 1, \dots, n$ , let  $i_h$  be the smallest index in  $[M_y, K]$  such that clock  $x_h$  is reset on the transition  $\rho[i_h - 1, i_h]$ . Assume without loss of generality that  $i_1 \leq i_2 \leq \dots \leq i_n$ . We set  $i_0 = M_y$  and  $i_{n+1} = K + 1$ . Thus, for every interval  $[i_h, i_{h+1} - 1]$ ,  $0 \leq h \leq n$ , the following holds: for all  $x \in X(P)$ , either clock  $x$  is never reset along  $\rho[i_h, i_{h+1} - 1]$  or its value is always less than  $N_A - c_A$ . Since for each parametric atomic constraint  $f = x' - x < e'$  of  $\mathcal{A}$ ,  $e'[v] \leq e'[v_A] < c_A - N_A$ , we have that  $(v, w) \models f$  implies  $w(x) > N_A - c_A$ . Hence, for every interval  $[i_h, i_{h+1} - 1]$  and  $x \in X(P) \setminus \{x_0\}$ , either clock  $x$  is never reset along  $\rho[i_h, i_{h+1} - 1]$ , or none of the parametric atomic constraints along  $\rho[i_h, i_{h+1} - 1]$  is of the form  $x' - x < e'$ . Since  $n + 1 \leq k_A$ ,  $N_A - c_A = k_A(N_{\mathcal{R}(\mathcal{A})} + 1)$ , and  $DUR(\rho[i_h - 1, i_h]) = 0$  for  $h = 1, \dots, n$  (i.e., the only transition of  $\rho[i_h - 1, i_h]$  is instantaneous), there is a  $k$  such that  $DUR(\rho[i_k, i_{k+1} - 1]) \geq N_{\mathcal{R}(\mathcal{A})} + 1$ . Recall that for each  $\tau \in \mathbb{N}$  there is  $i \geq M_y$  such that  $DUR(\rho[M_y, i]) = \tau$ , and  $N_{\mathcal{R}(\mathcal{A})}$  is the number of equivalence classes induced by  $\approx_{v_{null}}$ . Hence, there are indexes  $i, j \in [i_k, i_{k+1} - 1]$  such that  $DUR(\rho[i, j]) \geq 1$  and  $s_i \approx_{v_{null}} s_j$ . Therefore, the claim holds.  $\square$

Let  $[i, j] \subseteq [M_y, M_z]$  be an interval satisfying the above claim. We can apply Lemma 2 to  $\rho[i, j]$  obtaining a finite run  $\rho_1$  starting from  $s_j$  and leading to  $s'_j \sqsupseteq s_j$ . Thus we can repeatedly apply Lemma 1 to append an arbitrary number  $d$  of simulations of  $\rho_1$  and then simulate the remaining part of  $\rho$ . Let  $\rho' = \rho[0, j] \rho_1 \rho_2 \dots \rho_d \rho'_{M_z} \rho''$  be the obtained run, where for  $h = 2, \dots, d$ , runs  $\rho_h$  are the simulations of  $\rho_1$ ,  $\rho'_{M_z}$  is the simulation of  $\rho[j, M_z]$ , and  $\rho''$  is the simulation of the remaining suffix of  $\rho$ . Note that by Lemmas 1 and 2  $\rho'$  is an accepting infinite run of  $\llbracket \mathcal{A} \rrbracket_v$  and the clock constraint  $f_p$  never appears along  $\eta = \rho[0, j] \rho_1 \rho_2 \dots \rho_d \rho'_{M_z}$ , hence  $\eta$  is also a finite run of  $\llbracket \mathcal{A} \rrbracket_{v_p}$ . Moreover,  $DUR(\rho_h) \approx DUR(\rho[i, j])$  for  $h = 1, \dots, d$ , and  $y$  is not reset in  $\rho_1 \rho_2 \dots \rho_d \rho'_{M_z}$ .

Let  $s = (q, w)$  be the last state of  $\rho'_{M_z}$ . Since  $s \sqsupseteq s_{M_z}$  and  $w_{M_z}(z) = 0$ , we have  $w(z) = 0$ . Being  $DUR(\rho[i, j]) \geq 1$ , by carefully choosing  $d$ , we get that  $(v_p, w) \models f_p$ . Thus, if clock  $y$  is never reset along  $\rho''$ , then  $\rho''$  is also a run in  $\llbracket \mathcal{A} \rrbracket_{v_p}$ , hence  $\rho'$  is an infinite accepting run in  $\llbracket \mathcal{A} \rrbracket_{v_p}$ . Otherwise, there is a non empty prefix  $\pi$  of  $\rho''$  (containing some instantaneous transition) such that  $\rho[0, j] \rho_1 \rho_2 \dots \rho_d \rho'_{M_z} \pi$  is a run of  $\llbracket \mathcal{A} \rrbracket_{v_p}$  and the remaining suffix of  $\rho''$  starts at a state in which clock  $y$  is zero. By iterating the above reasoning (starting from  $\rho''$ ) we get an accepting run of  $\llbracket \mathcal{A} \rrbracket_{v_p}$ , and the theorem is proved.  $\square$

Since  $\Gamma(\mathcal{A})$  is downward-closed, by the above theorem checking universality reduces to check non-emptiness of the TA  $\mathcal{A}_{v_A}$ . Since the largest constant in  $\mathcal{A}_{v_A}$  is bounded by  $|P| \cdot N_A \cdot c_A$  and  $N_A = O(|Q| \cdot k_A \cdot (2c_A + 2)^{2|X|^2})$ , by Theorem 1 we obtain the following result.

**Theorem 4.** *Given a lower bound automaton  $\mathcal{A}$ , checking for the universality of  $\Gamma(\mathcal{A})$  is PSPACE-complete and can be done in time exponential in  $|X|^4$  and in the size of the encoding of  $c_A$ , and polynomial in the number of parameters and locations of  $\mathcal{A}$ .*

## 4 Decision Problems for L/U Automata

In this section, we briefly discuss our results concerning the other decision problems we have mentioned in the introduction. We start giving the results on emptiness and universality for upper bound automata. Next, we combine the results we have given for lower bound and upper bound automata to solve such problems for general L/U automata. Then, we extend the considered problems placing linear constraints on the parameters. Finally, we use L/U automata to decide satisfiability and model-checking related problems for a dense-time linear temporal logic.

**Upper bound automata.** The arguments used to show the results for upper bound automata are dual to those used for lower bound automata. We fix an upper bound automaton  $\mathcal{A} = \langle Q, q^0, \Delta, F \rangle$ . Recall that every linear expression of  $\mathcal{A}$  is of the form  $c_0 + c_1p_1 + \dots + c_m p_m$  with  $c_i \in \mathbb{N}$  for each  $1 \leq i \leq m$ . By increasing the parameter values, the clock constraints of  $\mathcal{A}$  are weakened, thus the set  $\Gamma(\mathcal{A})$  is upward-closed. An immediate consequence of this property is that testing universality of  $\Gamma(\mathcal{A})$  requires checking non-emptiness of the TA  $\mathcal{A}_{v_{null}}$  ( $v_{null}$  assigns 0 to each parameter). For checking emptiness of  $\Gamma(\mathcal{A})$ , we establish a version of Theorem 3 for upper bound automata. Here, we use a slightly larger constant  $N_{\mathcal{A}} = 8k_{\mathcal{A}}c_{\mathcal{A}}(N_{\mathcal{R}(\mathcal{A})} + 1) + c_{\mathcal{A}}$ . The definition of such constant is again motivated by counting arguments as in the case of lower bound automata. Define  $v_{\mathcal{A}}$  as the valuation assigning  $N_{\mathcal{A}}$  to each parameter.

**Theorem 5.** *Let  $v, v'$  be parameter valuations such that  $v \geq v' \geq v_{\mathcal{A}}$ . Then,  $v \in \Gamma(\mathcal{A})$  implies  $v' \in \Gamma(\mathcal{A})$ .*

Since  $\Gamma(\mathcal{A})$  is upward-closed, Theorem 5 implies that  $\Gamma(\mathcal{A})$  is not empty iff  $v_{\mathcal{A}} \in \Gamma(\mathcal{A})$ . Thus, checking emptiness of  $\Gamma(\mathcal{A})$  reduces to checking emptiness of the timed automaton  $\mathcal{A}_{v_{\mathcal{A}}}$ .

**General case.** Given an L/U automaton  $\mathcal{A}$ , if we instantiate the lower bound parameters of  $\mathcal{A}$ , we get an upper bound automaton and, similarly, if we instantiate the upper bound parameters of  $\mathcal{A}$ , we get a lower bound automaton. Furthermore, monotonicity properties continue to hold: if  $v \in \Gamma(\mathcal{A})$  and  $v'$  is such that  $v'(p) \leq v(p)$  for each lower bound parameter  $p$  and  $v'(p) \geq v(p)$  for each upper bound parameter  $p$ , then  $v' \in \Gamma(\mathcal{A})$ . By Theorems 3 and 5 it follows that

- To check for non-emptiness of  $\Gamma(\mathcal{A})$ , it suffices to check for non-emptiness of the timed automaton resulting from setting all the lower bound parameters to 0 and all the upper bound parameters to  $8k_{\mathcal{A}}c_{\mathcal{A}}(N_{\mathcal{R}(\mathcal{A})} + 1) + c_{\mathcal{A}}$ .
- To check for universality of  $\Gamma(\mathcal{A})$ , it suffices to check for non-emptiness of the timed automaton resulting from setting all the upper bound parameters to 0 and all the lower bound parameters to  $k_{\mathcal{A}}(N_{\mathcal{R}(\mathcal{A})} + 1) + c_{\mathcal{A}}$ .

Thus by Theorem 1 we obtain the following result.

**Theorem 6.** *For an L/U automaton  $\mathcal{A}$ , checking for the emptiness (resp. universality) of  $\Gamma(\mathcal{A})$  is PSPACE-complete and can be done in time exponential in  $|X|^4$  and the size of the encoding of  $c_{\mathcal{A}}$ , and polynomial in the number of parameters and locations of  $\mathcal{A}$ .*

**Linearly constrained parameters.** A linear constraint  $C$  is a boolean combination of inequalities and equations of the form  $e \sim 0$ , where  $e$  is a linear expression and  $\sim \in \{<, =\}$ . A parameter valuation  $v$  is a solution of  $C$  if the boolean expression obtained from  $C$  by replacing each inequality/equation  $e \sim 0$  with the truth value of  $e[v] \sim 0$ , evaluates to true. With  $Sol(C)$  we denote the set of  $C$  solutions. Given an L/U automaton  $\mathcal{A}$  and a linear constraint over the  $\mathcal{A}$  parameters, we consider the following decision problems:

- *Constrained emptiness:* given a constraint  $C$ , is the set  $\Gamma(\mathcal{A}) \cap Sol(C)$  empty?
- *Constrained universality:* given a constraint  $C$ , does  $\Gamma(\mathcal{A}) \supseteq Sol(C)$  hold?

We show that constrained emptiness and universality are decidable for both lower and upper bound automata. However, they become undecidable for L/U automata (the main reason being that a linear constraint can be used to force a lower bound parameter to be equal to an upper bound parameter, thus removing the restriction that has been placed on L/U automata). Decidability can be regained if we keep separated lower bound and upper bound parameters also in the linear constraint. In this case our approach relies on a bound for the set of *minimal* solutions of a linear constraint, given by Pottier [15], and our results on unconstrained emptiness and universality.

**Theorem 7.** *Constrained emptiness and constrained universality are undecidable for L/U automata. However, if we restrict to constraints where each equation/inequality is either over the set of lower bound parameters or over the set of upper bound parameters, then the problems are PSPACE-complete.*

**Parametric dense-time linear temporal logic.** We define the logic  $PMITL_{0,\infty}$  as a parametric extension of the logic  $MITL_{0,\infty}$  [4]. We impose a restriction on the use of parameters reflecting that imposed on the parameters of L/U automata (by [3], if we remove this restriction, then basic decision problems become undecidable). To this aim we fix two disjoint finite sets of parameters  $U$  and  $L$ , and denote with  $\mu$  (resp.,  $\lambda$ ) a linear expression over parameters  $U \cup L$  such that each parameter from  $U$  (resp.,  $L$ ) occurs positively and each parameter from  $L$  (resp.,  $U$ ) occurs negatively.

$PMITL_{0,\infty}$  formulas  $\varphi$  over a finite set  $AP$  of atomic propositions are defined as:

$$\varphi := a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \mathcal{U}_{<\mu} \varphi \mid \varphi \mathcal{U}_{>\lambda} \varphi \mid \varphi \mathcal{R}_{<\lambda} \varphi \mid \varphi \mathcal{R}_{>\mu} \varphi,$$

where  $a \in AP$ ,  $< \in \{\leq, <\}$ ,  $> \in \{\geq, >\}$  and  $\mathcal{U}_{<\mu}$  and  $\mathcal{U}_{>\lambda}$  (resp.,  $\mathcal{R}_{<\lambda}$  and  $\mathcal{R}_{>\mu}$ ) are the parameterized versions of the *until* modality (resp., *release* modality).

$PMITL_{0,\infty}$  is interpreted over *timed sequences* over  $2^{AP}$ , defined as infinite sequences  $\rho = (\sigma_0, I_0)(\sigma_1, I_1) \dots$ , where for all  $i$ ,  $\sigma_i \in 2^{AP}$ , and  $I_0, I_1, \dots$  represents a partition of  $\mathbb{R}_{\geq 0}$  in non-empty intervals such that for all  $i$ , the upper bound of  $I_i$  equals the lower bound of  $I_{i+1}$ . For  $t \in \mathbb{R}_{\geq 0}$ , let  $\rho(t)$  be the unique  $\sigma_i$  such that  $t \in I_i$ .

For a formula  $\varphi$ , a timed sequence  $\rho = (\sigma_0, I_0)(\sigma_1, I_1) \dots$ , a parameter valuation  $v$ , and  $t \in \mathbb{R}_{\geq 0}$ , the satisfaction relation  $(\rho, v, t) \models \varphi$  under valuation  $v$  is defined as follows (we omit the clauses for boolean connectives, which are standard).

- $(\rho, v, t) \models a \Leftrightarrow a \in \rho(t)$ ;
- $(\rho, v, t) \models \varphi \mathcal{U}_{<\mu} \psi \Leftrightarrow$  for some  $t' \geq t$  such that  $t' < \mu[v] + t$ ,  $(\rho, v, t') \models \psi$  and  $(\rho, v, t'') \models \varphi$  for all  $t \leq t'' < t'$ .
- $(\rho, v, t) \models \varphi \mathcal{U}_{>\lambda} \psi \Leftrightarrow$  for some  $t' > t + \lambda[v]$ ,  $(\rho, v, t') \models \psi$  and  $(\rho, v, t'') \models \varphi$  for all  $t \leq t'' < t'$ .
- $(\rho, v, t) \models \varphi \mathcal{R}_{<\lambda} \psi \Leftrightarrow$  for all  $t'$  such that  $t \leq t' < \lambda[v] + t$ , either  $(\rho, v, t') \models \psi$ , or  $(\rho, v, t'') \models \varphi$  for some  $t \leq t'' < t'$ ;
- $(\rho, v, t) \models \varphi \mathcal{R}_{>\mu} \psi \Leftrightarrow$  for all  $t' > \mu[v] + t$ , either  $(\rho, v, t') \models \psi$ , or  $(\rho, v, t'') \models \varphi$  for some  $t \leq t'' < t'$ .

For a formula  $\varphi$ , a timed sequence  $\rho$ , and a parameter valuation  $v$ ,  $\rho$  satisfies  $\varphi$  under valuation  $v$  if  $(\rho, v, 0) \models \varphi$ . Note that we have defined  $\text{PMITL}_{0,\infty}$  formulas in positive normal form. It is simple to verify that the until and the release operators are dual, and therefore, the logic is closed under semantic negation.

For such a logic, we study the related satisfiability and model-checking problems. For a given  $\text{PMITL}_{0,\infty}$  formula  $\varphi$  and an L/U automaton  $\mathcal{A}$  such that the lower (resp., upper) bound parameters of  $\mathcal{A}$  are from  $L$  (resp.,  $U$ ), we consider the emptiness and universality problems for the following sets of parameter valuations: the set  $S(\varphi)$  of parameter valuations that make  $\varphi$  satisfiable, and the set  $V(\mathcal{A}, \varphi)$  of parameter valuations  $v$  for which every timed sequence accepted by  $\llbracket \mathcal{A} \rrbracket_v$  satisfies  $\varphi$ . Note that the semantics of L/U automata can be slightly modified such that an L/U automaton recognizes timed sequences (see [4] for standard timed automata).

We solve the above decision problems by reducing them to corresponding problems on L/U automata. The key of these reductions is the translation of a  $\text{PMITL}_{0,\infty}$  formula into an equivalent L/U automaton. Such translation relies on the construction given in [4] for  $\text{MITL}_{0,\infty}$  and  $TA$ .

**Theorem 8.** *For a  $\text{PMITL}_{0,\infty}$  formula  $\varphi$  and an L/U automaton  $\mathcal{A}$ , checking for emptiness and universality of  $S(\varphi)$  and  $V(\mathcal{A}, \varphi)$  is PSPACE-complete.*

## 5 Conclusion

We have studied some decision problems on L/U automata. In particular, we have shown that the emptiness and universality problems for the set of parameter valuations for which there is an infinite accepting run are decidable and PSPACE-complete. This allows us to prove decidability of a parametric extension of  $\text{MITL}_{0,\infty}$ . Furthermore, we have studied a constrained version of emptiness and universality with parameters constrained by linear systems of equations and inequalities. For the ease of presentation we do not allow to specify clock invariants on locations of L/U automata. However, it is simple to verify that the addition of invariants would not change the validity of our arguments.

There are other results that can be derived from those presented here. As an example, we could combine the results on constrained decision problems along with those on  $\text{PMITL}_{0,\infty}$  to solve the constrained versions of the decision problems for  $\text{PMITL}_{0,\infty}$ . Moreover, when all the parameters in the model are of the same type (i.e., either lower bound or upper bound), it is possible to compute an explicit representation of the set  $\Gamma(\mathcal{A})$  by linear constraints over parameters (this can be done similarly to what is done



in [3] for PLTL). Also, we can solve some optimization problems on the parameter valuations, which can be very interesting for system designers, and decide the finiteness of the set  $\Gamma(\mathcal{A})$ .

As future research, we think of the extension of our results to real-valued parameters. The results we have shown in this paper answer only partially to this problem. Another interesting direction is to investigate the parametric extension of MITL [4], where constraints are expressed in form of intervals as opposed to bounds as in  $\text{PMITL}_{0,\infty}$ . The technique we have used here for  $\text{PMITL}_{0,\infty}$  does not seem to scale to such a logic, and a different approach may be required.

## References

1. Alur, R., Courcoubetis, C., Dill, D.L.: Model-checking in dense real-time. *Information and Computation* 104(1), 2–34 (1993)
2. Alur, R., Dill, D.: A theory of timed automata. *Theoretical Computer Science* 126(2), 183–235 (1994)
3. Alur, R., Etessami, K., La Torre, S., Peled, D.: Parametric temporal logic for model measuring. *ACM Transactions on Computational Logic* 2(3), 388–407 (2001)
4. Alur, R., Feder, T., Henzinger, Th.A.: The benefits of relaxing punctuality. *Journal of the ACM* 43(1), 116–146 (1996)
5. Alur, R., Henzinger, Th.A., Vardi, M.Y.: Parametric real-time reasoning. In: *Proc. of the 25th ACM Symposium on Theory of Computing (STOC'93)*, pp. 592–601. ACM Press, New York (1993)
6. Alur, R., Madhusudan, P.: Decision problems for timed automata: a survey. In: Bernardo, M., Corradini, F. (eds.) *Formal Methods for the Design of Real-Time Systems*. LNCS, vol. 3185, pp. 1–24. Springer, Heidelberg (2004)
7. Bruyère, V., Dall'Olio, E., Raskin, J.F.: Durations, Parametric Model-Checking in Timed Automata with Presburger Arithmetic. In: Alt, H., Habib, M. (eds.) *STACS 2003*. LNCS, vol. 2607, pp. 687–698. Springer, Heidelberg (2003)
8. Bruyère, V., Raskin, J.F.: Real-time model-checking: Parameters everywhere. In: Pandya, P.K., Radhakrishnan, J. (eds.) *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science*. LNCS, vol. 2914, pp. 100–111. Springer, Heidelberg (2003)
9. Emerson, E.A., Trefler, R.: Parametric Quantitative Temporal Reasoning. In: *Proc. 14th Ann. Symp. Logic in Computer Science (LICS'99)*, pp. 336–343. IEEE Computer Society Press, Los Alamitos (1999)
10. Henzinger, T., Kopke, P., Puri, A., Varaiya, P.: What's decidable about hybrid automata. *Journal of Computer and System Sciences* 57, 94–124 (1998)
11. Hune, T., Romijn, J., Stoelinga, M., Vaandrager, F.: Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming* 52,53, 183–220 (2002)
12. IEEE Computer Society. *IEEE Standard for a High Performance Serial Bus. Std 1394-1995* (August 1996)
13. Lamport, L.: A Fast Mutual Exclusion Algorithm. *ACM Transactions Computer Systems* 5(1), 1–11 (1987)
14. Pnueli, A.: The temporal logic of programs. In: *Proc. of the 18th IEEE Symposium on Foundations of Computer Science*, pp. 46–77. IEEE Computer Society Press, Los Alamitos (1977)
15. Pottier, L.: Minimal solutions of linear diophantine systems: bounds and algorithms. In: Book, R.V. (ed.) *Rewriting Techniques and Applications*. LNCS, vol. 488, pp. 162–173. Springer, Heidelberg (1991)
16. Wang, F.: Parametric timing analysis for real-time systems. *Information and Computation* 130(2), 131–150 (1996)

# On the Complexity of LTL Model-Checking of Recursive State Machines\*

Salvatore La Torre<sup>1</sup> and Gennaro Parlato<sup>1,2</sup>

<sup>1</sup>Università degli Studi di Salerno, Italy

<sup>2</sup>University of Illinois at Urbana-Champaign, USA

**Abstract.** Recursive state machines (RSMs) are models for programs with recursive procedural calls. While LTL model-checking is EXPTIME-complete on such models, on finite-state machines, it is PSPACE-complete in general and becomes NP-complete for interesting fragments. In this paper, we systematically study the computational complexity of model-checking RSMS against several syntactic fragments of LTL. Our main result shows that if in the specification we disallow *next* and *until*, and retain only the *box* and *diamond* operators, model-checking is in NP. Thus, differently from the full logic, for this fragment the abstract complexity of model-checking does not change moving from finite-state machines to RSMS. Our results on the other studied fragments confirm this trend, in the sense that, moving from finite-state machines to RSMS, the complexity of model-checking either rises from PSPACE-complete to EXPTIME-complete, or stays within NP.

## 1 Introduction

Linear temporal logic (LTL) is a specification language commonly used for expressing correctness requirements of reactive systems [10,11]. An LTL formula is built from atomic propositions, boolean connectives, and temporal modalities such as *next*, *diamond*, *box*, and *until*, and is interpreted over infinite sequences of states which model computations of reactive programs. A typical temporal requirement is “every request  $p$  is followed by a response  $q$ ,” and can be expressed in LTL as  $\Box ( p \rightarrow \Diamond q )$ .

Given an abstract model  $M$  of a reactive system and an LTL formula  $\varphi$ , LTL *model-checking* asks whether an infinite computation of  $M$  satisfying  $\varphi$  exists. When  $M$  is finite-state, this decision problem is PSPACE-complete [12]. Despite the high computational complexity, solutions to LTL model-checking are implemented in verification tools and successfully applied in practice (see [5,9]).

The control flow of programs with recursive procedure calls can be naturally captured with *recursive state machines* (RSMS). In an RSM, vertices can either be ordinary states or can correspond to invocations of other state machines in a potentially recursive manner. We recall that RSMS correspond to pushdown

---

\* This research was partially supported by the MIUR grant ex-60% 2005, Università degli Studi di Salerno.

systems, and the related model-checking problem has been extensively studied in the literature [14,8,13]. Among the many interesting results, we recall that LTL model-checking of RSMS is EXPTIME-complete [4].

On finite-state systems, the computational complexity of model-checking for meaningful LTL fragments improves (see [6]). In particular, for the logic of all formulae obtained by allowing an arbitrary nesting of *diamond* and *box* operators, which we denote with  $L(\diamond)$ , model-checking is NP-complete [12]. This fragment is very rich and can express many interesting properties of reactive systems, such as liveness (as the above response property) but also safety, fairness, and others.

In this paper, we systematically study the computational complexity of model-checking RSMS with respect to specifications from natural syntactic fragments of LTL. Our goal is the discovery of more efficient algorithms for interesting classes of specifications and a better understanding of the computational complexity of this decision problem. The main result of this paper concerns the computational complexity of model-checking RSMS with respect to  $L(\diamond)$  specifications. We show that this problem is in NP, that matches the known lower bound for finite-state machines. Thus, differently from full LTL, the computational complexity of  $L(\diamond)$  model-checking stays unchanged while moving from finite-state systems to RSMS. Considering the simple structure of  $L(\diamond)$  models [12], this result may not seem surprising. However, the effects of mixing this logic with the use of a stack were not completely clear. In the EXPTIME-hardness proof for LTL [4], the stack of the model is used to linearize computations of an alternating Turing machine working in polynomial space and this basically allows the authors to carry over the PSPACE-hardness proof of the LTL model-checking of finite-state machines. Moreover, the expressiveness of  $L(\diamond)$  combined with the alternation provided by a 2-player game graph suffices to prove a matching lower bound for deciding full LTL games [3].

To show NP membership for the model-checking of RSMS with respect to  $L(\diamond)$  specifications, we define a certificate (a finite sequence of alternate nodes and set of nodes of the RSM) and then show that the problem can be reduced to answering two distinct queries: (1) checking that the certificate satisfies the given formula, and (2) checking that there exists a run of the given RSM which matches the certificate. To complete the NP membership argument we also show that we only need to investigate certificates of length bounded by the size of the formula. Observe that the smallest model of an  $L(\diamond)$  formula among all the runs of an RSM can be of size over-polynomial, thus looking for an abstract representation of runs is needed and clearly the arguments used in showing NP membership of this problem on finite Kripke structures [12] cannot be applied here.

The other results of this paper concern the model-checking of RSMS with respect to specifications expressed in other syntactic fragments of LTL. For fragments whose model-checking problem on finite-state machines is PSPACE-complete, we show EXPTIME-hardness, and therefore the model-checking problem is EXPTIME-complete. For fragments whose model-checking problem on finite-state machines is NP-complete, we instead give an NP upper bound. Remarkably, for some of these fragments we re-use results shown to prove NP

membership for  $L(\diamond)$ . All the results are summarized in a table in the concluding section.

## 2 Logic and Models

### 2.1 Propositional Linear Temporal Logic

In this section, we briefly recall the logic LTL and LTL model-checking (see [7]).

Given a set of atomic propositions  $AP$ , a *propositional linear temporal logic* (LTL) formula is composed of atomic propositions from  $AP$ , the boolean connectives *negation* ( $\neg$ ), *conjunction* ( $\wedge$ ) and *disjunction* ( $\vee$ ), the temporal operators *next* ( $\bigcirc$ ), *diamond* ( $\diamond$ ), *box* ( $\square$ ), and *until* ( $\mathcal{U}$ ). Formulae are built up in the usual way from these operators and connectives, according to the following grammar

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \mathcal{U} \varphi,$$

where  $p$  is an atomic proposition of  $AP$ . The other formulae can be introduced as abbreviations in the usual way. In particular, the diamond operator is a restricted form of until (i.e.,  $\diamond\varphi \equiv \text{TRUE } \mathcal{U} \varphi$  holds), and the box operator expresses the logical negation of the diamond operator (i.e.,  $\diamond\varphi \equiv \neg(\square\neg\varphi)$  holds).

LTL formulae are interpreted on linear-time structures. A *linear-time structure* (or simply a *structure*) is a pair  $(\pi, \text{lab}_\pi)$  where  $\pi$  is an  $\omega$ -sequence  $\pi = s_0s_1s_2\dots$  of *states* and  $\text{lab}_\pi$  is a mapping  $\text{lab}_\pi : \{s_0, s_1, s_2, \dots\} \rightarrow 2^{AP}$  labeling each state  $s_i$  with the set of propositions that hold at  $s_i$ .

Let  $\pi$  be a structure,  $i \in \mathbb{N}_0$  be a position, and  $\varphi$  an LTL formula. The satisfaction relation  $\models$  is inductively defined as follows.

- $(\pi, i) \models p \stackrel{\text{def}}{\iff} p \in \text{lab}_\pi(s_i)$ ;
- $(\pi, i) \models \neg\varphi \stackrel{\text{def}}{\iff} (\pi, i) \not\models \varphi$  holds false (i.e.,  $(\pi, i) \not\models \varphi$ );
- $(\pi, i) \models \varphi \wedge \psi \stackrel{\text{def}}{\iff} (\pi, i) \models \varphi$  and  $(\pi, i) \models \psi$ ;
- $(\pi, i) \models \bigcirc\varphi \stackrel{\text{def}}{\iff} (\pi, i + 1) \models \varphi$ ;
- $(\pi, i) \models \varphi \mathcal{U} \psi \stackrel{\text{def}}{\iff} \exists j \geq i$  such that  $(\pi, j) \models \psi$ , and  $\forall k \in [i, j - 1], (\pi, k) \models \varphi$ ;

When  $(\pi, 0) \models \varphi$ , we also write  $\pi \models \varphi$  and say that  $\pi$  is a *model* of  $\varphi$ .

An LTL-formula  $\varphi$  is *satisfiable* if there exists a model of  $\varphi$ . Given an LTL formula  $\varphi$ , we denote with  $|\varphi|$  its length. We denote with  $L(\diamond)$  the LTL fragment that allows only  $\diamond$  and  $\square$  as temporal operators (no next or until operators are allowed).

Given a set of atomic propositions  $AP$ , a *Kripke structure* over  $AP$  is a tuple  $M = (S, \Delta, \text{lab}, I)$  where:  $S$  is a (possibly infinite) set of *states*;  $\Delta \subseteq S \times S$  is a *transition relation*;  $\text{lab} : S \rightarrow 2^{AP}$  is a *labeling function* that associates with each state  $s$  a set of atomic propositions  $\text{lab}(s)$  (meaning that the atomic propositions that hold true at  $s$  are exactly those in  $\text{lab}(s)$ );  $I \subseteq S$  is the set of the initial states of  $M$ .

We say that  $M$  is a *finite* Kripke structure when  $S$  is finite. A *run* or *path* of  $M$  is a (finite or infinite) sequence of states  $\pi = s_0s_1\dots$  of  $M$ , such that  $s_0 \in I$

and  $(s_i, s_{i+1})$  is a transition of  $M$ , with  $i \geq 0$ . Notice that, each run in  $M$  is a linear-time structure. We write  $M \models \varphi$  when there exists an infinite run  $\pi$  of  $M$  such that  $\pi \models \varphi$ . The *model-checking* problem is: “given a Kripke structure  $M$  and an LTL formula  $\varphi$ , does  $M \models \varphi$ ?” With  $|M|$  we denote the size of  $M$ , that is  $|S| + |\Delta|$ .

In the following, we use the result.

**Theorem 1 ([12]).** *The model-checking problem of a finite Kripke structure against an  $L(\diamond)$  formula is NP-complete.*

## 2.2 Recursive State Machines

In this section, we recall the recursive state machines (RSMs) introduced in [1].

*Syntax.* Let  $I_{\mathcal{M}} = \{1, \dots, k\}$ . A *recursive state machine* (RSM)  $\mathcal{M}$  over a set of atomic propositions  $AP$  is a tuple  $(M_1, \dots, M_k)$ , where  $M_h = (N_h, B_h, Y_h, En_h, Ex_h, E_h, true_h)$ , for  $h \in I_{\mathcal{M}}$ , is a module and consists of the following components:

- A finite nonempty set of *nodes*  $N_h$ , and a finite set of *boxes*  $B_h$ .
- A labeling  $Y_h : B_h \rightarrow I_{\mathcal{M}}$  that assigns to every box a module index.
- A set of *entry* nodes  $En_h \subseteq N_h$ , and a set of *exit* nodes  $Ex_h \subseteq N_h$ .
- Let  $Calls_h = \{(b, e) \mid b \in B_h, e \in En_j, j = Y_h(b)\}$  denote the set of *calls* of  $M_h$ , and  $Retns_h = \{(b, x) \mid b \in B_h, x \in Ex_j, j = Y_h(b)\}$  denote the set of *returns* in  $M_h$ . Then,  $E_h \subseteq ((N_h \cup Retns_h) \times (N_h \cup Calls_h))$  is the set of *edges* of  $M_h$ .
- A *labeling function*  $true_h : N_h \rightarrow 2^{AP}$  that associates a set of atomic propositions to each node of  $M_h$ .

We assume that  $N_1, \dots, N_k$  (respectively,  $B_1, \dots, B_k$ ) are pairwise disjoint. We define with  $N = \bigcup_{i \in I_{\mathcal{M}}} N_i$  the set of all *nodes* of  $\mathcal{M}$ , with  $B = \bigcup_{i \in I_{\mathcal{M}}} B_i$  the set of all *boxes* of  $\mathcal{M}$ , and with  $E = \bigcup_{i \in I_{\mathcal{M}}} E_i$  the set of all *edges* of  $\mathcal{M}$ . Also,  $Y : B \rightarrow I_{\mathcal{M}}$  denotes the extension of all the functions  $Y_h, h \in I_{\mathcal{M}}$ , and  $true : N \rightarrow 2^{AP}$  denotes the extension of all the functions  $true_h (h \in I_{\mathcal{M}})$ . Module  $M_k$  is the *initial* module of  $\mathcal{M}$ .

Fig. 1 illustrates the definition. The RSM has two modules. The module  $M_1$  has 5 nodes, of which  $e_1$  and  $e_2$  are the entry nodes and  $x_1$  and  $x_2$  are the exit nodes, and two boxes, of which  $b_1$  is mapped to module  $M_2$  and  $b_2$  is mapped to  $M_1$ . The entry and exit nodes are the control interface of a module by which it can communicate with the other modules. Intuitively, think of modules as procedures, and an edge entering a box invoking the procedure associated with the box.

*Semantics.* With each module of an RSM, we associate a Kripke structure by recursively substituting each box by the module referenced by the box. Formally, we associate to the module  $M_h$  the Kripke structure  $M_h^F = (S_h, \Delta_h, lab_h, I_h)$ , called the *expansion* or the *flat* of  $M_h$  in  $\mathcal{M}$ , as follows. The states  $S_h$  of  $M_h^F$  are elements of the form  $\langle \alpha u \rangle$  where  $\alpha u \in B^* \times N$  and either

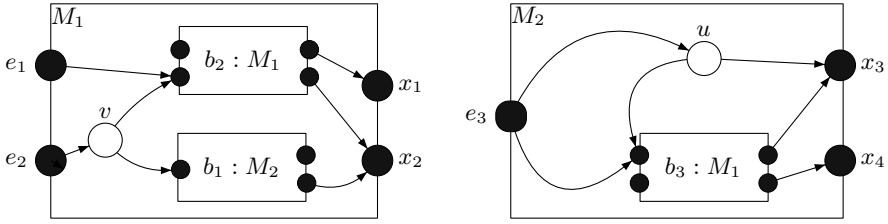


Fig. 1. A sample recursive state machine

- $\alpha = \epsilon$  and  $u \in N_h$ , or,
- $\alpha = b_1 \dots b_\ell$  (with  $\ell \geq 1$ ),  $b_1 \in B_h$ ,  $b_{i+1} \in B_{Y(b_i)}$  for every  $i \in [1, \ell - 1]$ , and  $u \in N_{Y(b_\ell)}$ .

The initial states of  $M_h^F$  are  $I_h = \{\langle e \rangle \mid e \in En_h\}$ . The labeling function is  $lab_h(\langle \alpha u \rangle) = true(u)$ . In the following, for the ease of presentation, we also denote the states of  $\mathcal{M}$  of the form  $\langle \alpha bu \rangle$ , where  $(b, u)$  is a call or a return, as  $\langle \alpha(b, u) \rangle$ . Therefore, states corresponding to calls or returns will have two representations and of the two we will always use the more convenient one, as for the following definition. Given two states  $X, X'$  of  $M_h^F$ , we define the transition relation  $\Delta_h$  as:  $(X, X') \in \Delta_h$  iff  $X = \langle \alpha u \rangle$ ,  $X' = \langle \alpha v \rangle$ , and  $(u, v)$  is an edge of  $\mathcal{M}$ . (Note that in order to make the above definition consistent, we implicitly use the representation of the form  $\langle \beta(b, z) \rangle$  for  $X$ , if  $(b, z)$  is a return, and for  $X'$ , if  $(b, z)$  is a call, and the standard representation in all the other cases.) The expansion  $M_k^F$  of the initial module of  $\mathcal{M}$  is also denoted  $\mathcal{M}^F$ .

Given a state  $X = \langle \alpha u \rangle$  with  $u \in N$ , we denote with  $node(X)$  the node  $u$ , and say that  $u$  is the *node* of  $X$ .

An RSM  $\mathcal{M}$  satisfies an LTL formula  $\varphi$ , written  $\mathcal{M} \models \varphi$ , iff  $\mathcal{M}^F \models \varphi$ . The model-checking problem of an RSM  $\mathcal{M}$  against an LTL formula  $\varphi$  is to determine whether  $\mathcal{M}$  satisfies  $\varphi$ . The following result holds.

**Theorem 2 ([1]).** *The problem of model-checking an RSM against an LTL formula is EXPTIME-complete.*

### 3 The Complexity of $L(\diamond)$ Model-Checking on RSMS

In this section we show that model-checking of  $L(\diamond)$  formulae against RSMS is NP-complete. We prove first some preliminary results and then use them to argue the claimed result.

Fix an RSM  $\mathcal{M}$  and an  $L(\diamond)$  formula  $\varphi$ . Our first goal is to define a finite certificate for a run that will be used in the argument to show membership in NP. An  $\mathcal{M}$ -sequence of length  $\ell$  is a sequence  $\gamma = u_0, U_1, u_2, U_3, \dots, u_{2\ell}, U_{2\ell+1}$  such that  $u_0, u_2, \dots, u_{2\ell}$  are  $\mathcal{M}$  nodes, and  $U_1, U_3, \dots, U_{2\ell+1}$  are sets of  $\mathcal{M}$  nodes. For a run  $\pi$  of  $\mathcal{M}^F$  and an  $\mathcal{M}$ -sequence  $\gamma$  as above, we say that  $\pi$  and  $\gamma$  match if  $\pi$  can be factored as  $X_0\pi_1X_2\pi_3 \dots X_{2\ell}\pi_{2\ell+1}$  where  $X_{2i}$  are states,  $\pi_{2i+1}$  are

runs,  $node(X_{2i}) = u_{2i}$ , and all the nodes of states occurring in  $\pi_{2i+1}$  belong to  $U_{2i+1}$ , for every  $i \in [0, \ell]$ , and for all  $u \in U_{2\ell+1}$ ,  $u$  is the node of infinitely many states in  $\pi_{2\ell+1}$ .

It is crucial for our argument to be able to decide in polynomial time whether there exists an  $\mathcal{M}^F$  run that matches a given  $\mathcal{M}$ -sequence  $\gamma$ . We can construct a simple Büchi automaton  $A$  of size linear in the sizes of  $\mathcal{M}$  and  $\gamma$  which accepts infinite sequences matching  $\gamma$ . Therefore, the above problem reduces to checking for emptiness the intersection of  $\mathcal{M}$  and  $A$ . Thus, by the results shown in [118] we have:

**Lemma 1.** *Given an  $\mathcal{M}$ -sequence  $\gamma$ , deciding if there exists a run of  $\mathcal{M}^F$  matching  $\gamma$  can be done in deterministic polynomial time in the size of  $\mathcal{M}$  and the length of  $\gamma$ .*

The second important step in our argument for showing NP membership consists of defining a notion of satisfiability of formulae on  $\mathcal{M}$ -sequences that can be checked efficiently and that is sufficient to show satisfiability on a matching run of  $\mathcal{M}^F$ .

Fix an  $\mathcal{M}$ -sequence  $\gamma = u_0, U_1, u_2, U_3, \dots, u_{2\ell}, U_{2\ell+1}$ . We denote with  $BD(\varphi)$  the set of all the box/diamond sub-formulae  $\psi$  of  $\varphi$ , and for a linear structure  $\pi = s_0s_1s_2\dots$ , we denote with  $BD_\pi^\varphi(s_i)$  the set of all formulae  $\psi$  of  $BD(\varphi)$  such that  $(\pi, i) \models \psi$ .

We say that  $\gamma$  satisfies  $\varphi$  if:

1. the sequence  $\pi = u_0u_2\dots u_{2\ell}\pi'$  is a model of  $\varphi$ , where  $\pi'$  is any infinite sequence over  $U_{2\ell+1}$  such that each  $u \in U_{2\ell+1}$  occurs as the node of infinitely many states of  $\pi'$ ;
2. for every  $i \in [1, \ell]$  and for every  $u$  in  $U_{2i-1}$ ,  $BD_\pi^\varphi(u) = BD_\pi^\varphi(u_{2i})$  where  $\hat{\pi} = uu_{2i}\dots u_{2\ell}\pi'$ .

The following proposition elaborates on some results shown in [112] for the logic we are considering.

**Proposition 1.** *Let  $\pi = s_0s_1s_2\dots$  be a structure and  $\pi'$  be the suffix of  $\pi$  such that for each state  $s$  in  $\pi'$  there are infinitely many states  $s'$  in  $\pi'$  such that  $lab_\pi(s) = lab_\pi(s')$  □*

1. For an  $L(\diamond)$  formula  $\varphi$ , the set of  $\varphi$  sub-formulae that hold at a state  $s_i$  depends only on the atomic propositions that hold at  $s_i$  and the box/diamond sub-formulae that hold at  $s_{i+1}$  along  $\pi$ .
2. For  $\varphi \in L(\diamond)$ ,  $BD_\pi^\varphi(s') = BD_\pi^\varphi(s'')$  for every pair of states  $s', s''$  within  $\pi'$ .

The following lemma states that satisfaction of a formula  $\varphi$  on a given  $\mathcal{M}$ -sequence can be efficiently verified.

**Lemma 2.** *Given an  $L(\diamond)$  formula  $\varphi$  and an  $\mathcal{M}$ -sequence  $\gamma$ , checking that  $\gamma$  satisfies  $\varphi$  can be done in polynomial time.*

---

<sup>1</sup> Note that such a suffix of  $\pi$  always exists since  $2^{AP}$  is finite.

*Proof.* Checking part 1 of the definition of satisfiability on  $\mathcal{M}$ -sequences is trivial and can be done in linear time (see [12]). From Proposition 1, we can start computing  $BD_\pi^\varphi(u)$  for a  $u$  in  $\pi'$ , and then we compute the subsets  $BD_\pi^\varphi(u_{2i})$  for  $i = \ell, \dots, 0$ . Therefore, the lemma is proved.  $\square$

A crucial step in our argument for showing membership in NP is to prove that we can answer to our model-checking question simply searching for an  $\mathcal{M}$ -sequence  $\gamma$  which satisfies the given formula and checking for the existence of a run of  $\mathcal{M}^F$  matching  $\gamma$ . The next two lemmas prepare such a result.

**Lemma 3.** *If an  $\mathcal{M}$ -sequence  $\gamma$  satisfies  $\varphi$  then each run of  $\mathcal{M}^F$  which matches  $\gamma$  also satisfies  $\varphi$ .*

*Proof.* Let  $\gamma = u_0, U_1, u_2, \dots, U_{2\ell-1}, u_{2\ell}, U_{2\ell+1}$  be an  $\mathcal{M}$  sequence satisfying  $\varphi$  and  $\pi$  be a run of  $\mathcal{M}^F$  that matches  $\gamma$ . Thus,  $\pi$  can be factorized as  $X_0\pi_1X_2\pi_3 \dots X_{2\ell}\pi_{2\ell+1}$  where for all  $u \in U_{2\ell+1}$ ,  $u$  occurs infinitely often in  $\pi_{2\ell+1}$  and for  $i \in [0, \ell]$ :  $X_{2i}$  are states,  $\pi_{2i+1}$  are runs,  $node(X_{2i}) = u_{2i}$ , and all the nodes of  $\pi_{2i+1}$  states belong to  $U_{2i+1}$ . By part 2 of Proposition 1, we have that we can compute the set  $BD_\pi^\varphi(X)$  for each state  $X$  of  $\pi_{2\ell+1}$  by simply computing it for one of them. Since the sets of atomic propositions which occur infinitely often in  $\pi_{2\ell+1}$  are exactly the sets of atomic propositions which label the nodes of  $U_{2\ell+1}$ , by part 1 of Proposition 1,  $BD_\pi^\varphi(u_0) = BD_\pi^\varphi(X_0)$  holds. Therefore,  $\pi$  satisfies  $\varphi$  and the lemma is proved.  $\square$

**Lemma 4.** *If a run  $\pi$  of  $\mathcal{M}^F$  satisfies  $\varphi$  then there is an  $\mathcal{M}$ -sequence of length  $\ell \leq |BD(\varphi)| + 1$  which satisfies  $\varphi$  and matches  $\pi$ .*

*Proof.* Let  $\pi = X_0X_1X_2 \dots$  be a run of  $\mathcal{M}^F$  that satisfies  $\varphi$ . Denote with  $\rho$  the sequence  $s_0s_1s_2 \dots$  where  $s_i = node(X_i)$  for all  $i = 0, 1, 2, \dots$ . We construct a matching  $\mathcal{M}$ -sequence  $\gamma$  as follows. For each diamond sub-formula  $\psi$  of  $\varphi$ , consider the maximum index  $i$  such that  $(\rho, i) \models \psi$  (if defined). Instead, for each box sub-formula  $\psi$  of  $\varphi$ , consider the maximum index  $i$  such that  $(\rho, i) \not\models \psi$  (if defined). Take the nodes corresponding to such indices and denote them as  $u_2, \dots, u_{2\ell-2}$  (by increasing indices). Observe that  $\ell \leq |BD(\varphi)| + 1$ . Since the number of  $\mathcal{M}$  nodes is finite, there is a suffix  $\rho'$  of  $\rho$  starting after  $u_{2\ell-2}$  such that all nodes of  $\rho'$  occur infinitely often in  $\rho'$ . Take as node  $u_{2\ell}$  an arbitrary node in  $\rho'$  and as  $u_0$  the first node of  $\rho$ . Therefore,  $\rho$  can be factorized as  $u_0\rho_1u_2 \dots \rho_{2\ell-1}u_{2\ell}\rho_{2\ell+1}$ . (Note that  $\rho_{2\ell+1}$  is a suffix of  $\rho'$ .) Thus, we define  $\gamma$  as  $u_0, U_1, u_2, \dots, U_{2\ell-1}, u_{2\ell}, U_{2\ell+1}$ , where  $U_{2i+1}$  is the set of all nodes of  $\rho_{2i+1}$  for  $i = 0, \dots, \ell$ . By construction,  $\gamma$  satisfies  $\varphi$ , and the lemma is shown.  $\square$

By Lemmas 3 and 4, we get the following theorem.

**Theorem 3.**  *$\mathcal{M} \models \varphi$  if and only if there are an  $\mathcal{M}$ -sequence  $\gamma$  of length at most  $|BD(\varphi)| + 1$  and an  $\mathcal{M}^F$  run  $\pi$  such that  $\pi$  and  $\gamma$  match, and  $\gamma$  satisfies  $\varphi$ .*

Now we are ready to show membership in NP for the model-checking of  $L(\diamond)$  formulae on RSMs.



**Lemma 5.**  $L(\diamond)$  model-checking on RSMs is in NP.

*Proof.* Let  $\mathcal{M}$  be an RSM and  $\varphi$  be a formula. To check in nondeterministic polynomial time if there is a run of  $\mathcal{M}^F$  satisfying  $\varphi$  we can guess an  $\mathcal{M}$ -sequence  $\gamma$  of length at most  $|BD(\varphi)| + 1$ , then check if it satisfies  $\varphi$  and if there exists a run of  $\mathcal{M}^F$  which matches it. The correctness and the time complexity of this algorithm are a consequence of Lemmas 1 and 2, and Theorem 3.  $\square$

From Lemma 5 and from the fact that  $L(\diamond)$  model-checking is NP-hard already on finite Kripke structures (see [12]), we have the main result of this section.

**Theorem 4.**  $L(\diamond)$  model-checking on RSMs is NP-complete.

## 4 Syntactic Fragments of LTL

In this section, we show the complexity bounds on other natural syntactic fragments of LTL. We start introducing a notation for systematically defining such logics.

We denote with  $L(H_1, H_2, \dots)$  the fragment which allows only the temporal operators  $H_1, H_2, \dots$ . For instance  $L(\mathcal{U})$  is “LTL without  $\circ$ ”. Consistently, we have denoted with  $L(\diamond)$  the fragment with only  $\diamond$  and  $\square$ . For a formula  $\varphi$ , the *temporal height* of  $\varphi$  is the maximum number of nested temporal operators in it. We write  $L^k(H_1, \dots)$  to denote the fragment of  $L(H_1, \dots)$  where  $k \geq 0$  corresponds to the maximum temporal height which is allowed. We omit the index  $k$  when no bound is imposed, i.e.  $L(H_1, \dots) = L^\omega(H_1, \dots)$ .

### Formulae of Temporal Height 1

The logic  $L^1(\mathcal{U}, \circ)$  contains all LTL formulae of temporal height 1, that is, all such formulae where temporal operators are not nested.

Model-checking RSMs against formulae of  $L^1(\mathcal{U}, \circ)$  is NP-complete. To show this we observe that since temporal height is 1, formulae of this fragment are boolean combinations of simple formulae with just one temporal operator. Formulae which use only the until operator without nesting it have simple properties as those shown for  $L(\diamond)$  formulae in Section 3. Therefore, we can handle boolean combinations of such formulae as we have done for formulae of  $L(\diamond)$ . Being the next operator non nested, we can only use it to specify properties on the second state of a run, and therefore we get membership to NP for  $L^1(\mathcal{U}, \circ)$ . Since  $L^1(\diamond)$  is already NP-hard for Kripke structures [12], we have the following theorem.

**Theorem 5.**  $L^1(\mathcal{U}, \circ)$  model-checking of RSMs is NP-complete.

As a corollary of the above result and since  $L^1(\diamond)$  is NP-hard, we get:

**Corollary 1.** Model-checking of RSMs for the logics  $L^1(\diamond, \circ)$  and  $L(\mathcal{U})$  is NP-complete.

### Fragments with Only “Until” Formulae

In this section, we show that model-checking of RSMs is already EXPTIME-hard for  $L^2(\mathcal{U})$  formulae, and thus for the fragments  $L^{2+k}(\mathcal{U})$ ,  $L^{2+k}(\mathcal{U}, \circ)$  and  $L(\mathcal{U})$ . We adapt the proof given in [4] to show hardness for full LTL. We briefly sketch the differences with the original proof and ask the reader to refer to [4] for further details.

**Theorem 6.**  *$L^2(\mathcal{U})$  model-checking of RSMs is EXPTIME-complete.*

*Proof.* Recall that the proof given in [4] consists of a reduction from the membership problem for linearly bounded alternating Turing machines. Given an alternating Turing machine  $M$  and an input word  $w$ , the reduction constructs a pushdown system  $A$  and an  $L(\diamond, \circ)$  formula  $\varphi$  such that  $M$  accepts  $w$  if and only if there exists a sequence  $x$  accepted by  $A$  such that  $x \models \varphi$ . The idea is that  $x$  encodes a computation of  $M$  starting with  $w$  on the input tape. The formula  $\varphi$  is mainly responsible for checking consistency between consecutive configurations. The automaton  $A$  takes care of the tree structure of  $M$  computations and ensures that consecutive configurations of the computation tree are encoded consecutively in the word  $x$ .

The word  $x$  encodes configurations position by position and consecutive configurations are separated by symbol  $\#$ . Formula  $\varphi$  has the form

$$\Box((p_{\#} \wedge \bigcirc^{n+2} p_{\#}) \Rightarrow (\varphi_1 \wedge \varphi_2 \wedge \varphi_3))$$

where  $\varphi_1$ ,  $\varphi_2$  and  $\varphi_3$  are boolean combinations involving sub-formulae of the form  $\bigcirc^i \psi$  or simple atomic propositions. Note that the condition  $(p_{\#} \wedge \bigcirc^{n+2} p_{\#})$  holds true on the beginning of each configuration except the last one. We aim to show that we can replace every sub-formula of the form  $\bigcirc^i \psi$  with an until formula, and therefore the reduction will hold for also for the fragment  $L^2(\mathcal{U})$ .

For such purpose, let  $0, 1, 2$  be fresh atomic propositions. We label all the positions of a configuration with an atomic proposition from  $\{0, 1, 2\}$  and require that the first configuration in  $x$  is marked with  $0$ , the second with  $1$ , the third with  $2$ , the fourth with  $0$ , and so on. We also require that each position can be distinguished by the other positions in the configuration (i.e., we use pairwise disjoint sets of atomic propositions for encoding the content of each position). We use superscripts on the atomic propositions to denote the position in the configuration.

Consider a  $\varphi$  sub-formula  $\bigcirc^j p_a$ ,  $j \leq n$ , we recall that according to the encoding from [4], the purpose of this formula is to check that the  $j$ -th position of the current configuration contains symbol  $a$ . We substitute this formula with  $\bigwedge_{i=0}^2 (i \rightarrow (i \mathcal{U} p_a^j))$ .

Consider now a  $\varphi$  sub-formula  $\bigcirc^{n+j+2} p_a$ ,  $j \leq n$ , we recall that the purpose of this formula is to check that the  $j$ -th position of the next configuration contains symbol  $a$ . We substitute this formula with  $\bigwedge_{i=0}^2 (i \rightarrow ((i \vee (i+1)) \mathcal{U} (p_a^j \wedge (i+1))))$ , where  $(i+1)$  is modulo 3.

Note that there are no sub-formulae of  $\varphi$  that relate configurations that are not consecutive, i.e., the index  $j$  of a sub-formula  $\bigcirc^j p_a$  of  $\varphi$  is not larger than  $2n+2$ .

Moreover, we do not use more than an until operator for each maximal subformula of nested next operators. Therefore, the formula obtained by translating  $\varphi$  in the above way has temporal height 2. Besides, the size of the new formula is linear in the size of  $\varphi$ . Thus, we have the theorem.  $\square$

Thus, from Theorems 2 and 6, we have the following corollary

**Corollary 2.** *Model-checking of RSMs for the logics  $L^{2+k}(\mathcal{U})$ ,  $L^{2+k}(\mathcal{U}, \circ)$  and  $L(\mathcal{U})$ , for each  $k$ , is EXPTIME-complete.*

## The Complexity of Nesting “Next” Operator

We consider first the logic  $L^k(\diamond, \circ)$ , that is, the logic containing only formulae over the boolean operators and the temporal operators diamond and next, whose temporal height is bounded by  $k$ .

For a formula  $\varphi$  in such fragment we can push the next operators on the atomic propositions with a quadratic blow up. Then, we can replace each maximal next sub-formula with a fresh proposition thus obtaining an  $L^k(\diamond)$  formula  $\varphi'$  of quadratic size in  $|\varphi|$ . For a given RSM  $\mathcal{M} = (M_1, \dots, M_h)$  we consider a new RSM  $\mathcal{M}'$  such that for each machine  $M_i$  we add a machine  $M_i^{v_1 \dots v_k}$  for each sequence  $v_1 \dots v_k$  of  $\mathcal{M}$  nodes. Each such machine simulates  $M_i$  and maintains in the control the tuple of the next  $k$  nodes. Thus nodes, boxes and transitions are added consistently with this meaning. We label each node with the atomic propositions of the corresponding node of  $\mathcal{M}$  and with the new atomic propositions corresponding to the next formulae that are satisfied on the sequence of the next  $k$  nodes which is paired with it. Observe that since  $k$  is constant the size of  $\mathcal{M}'$  is polynomial in the size of  $\mathcal{M}$ . Therefore, model-checking  $\varphi$  on  $\mathcal{M}$  reduces in polynomial time to model-checking  $\varphi'$  on  $\mathcal{M}'$ . Observe that, on formulae of  $L^k(\diamond)$ , for each fixed  $k$ , we can determinize the nondeterministic algorithm we have given in the NP membership proof of  $L(\diamond)$  such that the resulting algorithm runs in polynomial time (recall  $k$  is constant). Therefore we have the following theorem.

**Theorem 7.**  *$L^k(\diamond, \circ)$  model-checking of RSMs is in PTIME.*

If we allow nesting the next operator an unbounded number of times, then the model-checking problem becomes NP-complete. Membership in NP is trivial: we just guess a sequence of  $k$  nodes and simulate the RSM according to this sequence. NP-hardness is inherited from the complexity of model-checking the same fragment of LTL on finite Kripke structures [6].

**Theorem 8.**  *$L(\circ)$  model-checking of RSMs is NP-complete.*

## 5 Conclusion

In this paper we have analyzed the computational complexity of model-checking fragments of LTL by restricting the choice of the temporal operators and the

**Table 1.** Model-checking of RSMS against LTL fragments by bounding the temporal height

| $0 \leq k < \omega$              | Recursive State Machines | Finite Kripke Structures |
|----------------------------------|--------------------------|--------------------------|
| $L(\diamond)$                    | NP-complete              | NP-complete [12]         |
| $L^{1+k}(\diamond)$              | NP-complete              | NP-complete [12]         |
| $L(\diamond, \bigcirc)$          | EXPTIME-complete [4]     | PSPACE-complete [12]     |
| $L^{1+k}(\diamond, \bigcirc)$    | NP-complete              | NP-complete [6,12]       |
| $L(\mathcal{U})$                 | EXPTIME-complete         | PSPACE-complete [12]     |
| $L^{2+k}(\mathcal{U})$           | EXPTIME-complete         | PSPACE-complete [6]      |
| $L^1(\mathcal{U})$               | NP-complete              | NP-complete [6]          |
| $L(\bigcirc)$                    | NP-complete              | NP-complete [6]          |
| $L^k(\bigcirc)$                  | PTIME                    | PTIME [6]                |
| $L(\mathcal{U}, \bigcirc)$       | EXPTIME-complete [4]     | PSPACE-complete [12]     |
| $L^{2+k}(\mathcal{U}, \bigcirc)$ | EXPTIME-complete         | PSPACE-complete [6]      |
| $L^1(\mathcal{U}, \bigcirc)$     | NP-complete              | NP-complete [6]          |

temporal height of formulae. The complete picture of the complexity results for the considered LTL fragments is summarized in Table 1. Observe that the model-checking of fragments that are PSPACE-complete on finite Kripke structures becomes EXPTIME-complete on RSMS, instead for all the remaining fragments this problem stays in the same complexity class on either RSMS or finite Kripke structures.

Our main result is showing that  $L(\diamond)$  is in NP. The argument we use relies on two main properties: the ability of solving in polynomial time a variation of the reachability problem on the RSMS and the possibility of having a small representation for the instances of such problem which we need to check. The first result is not surprising, though it requires some expertise, since the set of reachable configurations in a pushdown system is regular and thus reachability-like queries can be efficiently computed. For the reasons discussed in the introduction, what was not clear was the kind of certificate to use in our arguments and also if a “small” certificate existed for such problem.

In [1], it is shown that LTL model-checking for RSMS and for pushdown systems are inter-reducible in linear time. Moreover, the acceptance problem of linearly bounded alternating Turing machines can be reduced to LTL model-checking on pushdown systems (see [4]). Thus, the EXPTIME-completeness of LTL model-checking problem for RSMS comes directly from pushdown systems. Actually, the proof given in [4] also constitutes a proof of EXPTIME-hardness for  $L(\diamond, \bigcirc)$ .

Investigating logics of formulae with bounded temporal height is interesting and gives a better understanding of the actual complexity of real instances of the model-checking problem. In fact, in the applications, the temporal height often turns out to be at most 2 (or 3 when fairness is involved) even when the specification is quite large and combines a large number of temporal constraints. A bounded height is often invoked as a reason why LTL model-checking is feasible in practice [6]. We think that such investigation for pushdown systems is even

more interesting because of the extremely high complexity of the problem for full LTL.

The logic LTL does not allow to express non regular properties on the runs of an RSM, such as properties of the call-stack. The logic CARET [2] extends LTL with operators that allow for example to check the call-stack content or express properties on the effects of runs within modules (local properties). Model-checking CARET specifications is also EXPTIME-complete. An interesting line of future research is to systematically study the impact of the CARET operators on the complexity of model-checking. It is simple to verify that, except for the operators that refer to the call-stack content, there are no changes with what we have shown in this paper. Remarkably, we can show that model-checking formulae that allow to inspect the stack sequence using the diamond operator is PSPACE-hard, and we conjecture that is also in PSPACE.

## References

1. Alur, R., Benedikt, M., Etessami, K., Godefroid, P., Reps, T.W., Yannakakis, M.: Analysis of recursive state machines. *ACM Trans. Program. Lang. Syst.* 27(4), 786–818 (2005)
2. Alur, R., Etessami, K., Madhusudan, P.: A temporal logic of nested calls and returns. In: Jensen, K., Podelski, A. (eds.) *TACAS 2004*. LNCS, vol. 2988, pp. 467–481. Springer, Heidelberg (2004)
3. Alur, R., Torre, S.L., Madhusudan, P.: Playing games with boxes and diamonds. In: Amadio, R.M., Lugiez, D. (eds.) *CONCUR 2003*. LNCS, vol. 2761, pp. 127–141. Springer, Heidelberg (2003)
4. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Application to model-checking. In: Mazurkiewicz, A. W, Winkowski, J. (eds.) *CONCUR 1997*. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
5. Clarke, E., Kurshan, R.: Computer-aided verification. *IEEE Spectrum* 33(6), 61–67 (1996)
6. Demri, S., Schnoebelen, P.: The complexity of propositional linear temporal logics in simple cases. *Inf. Comput.* 174(1), 84–103 (2002)
7. Emerson, E.A.: Temporal and modal logic. In: *Handbook of Theoretical Computer Science. Formal Models and Semantics (B)*, vol. B, pp. 995–1072 (1990)
8. Esparza, J., Hansel, D., Rossmanith, P., Schwoon, S.: Efficient algorithms for model checking pushdown systems. In: Emerson, E.A., Sistla, A.P. (eds.) *CAV 2000*. LNCS, vol. 1855, pp. 232–247. Springer, Heidelberg (2000)
9. Holzmann, G.J.: The model checker spin. *IEEE Trans. Software Eng.* 23(5), 279–295 (1997)
10. Manna, Z., Pnueli, A.: *The temporal logic of reactive and concurrent systems: Specification*. Springer, Heidelberg (1991)
11. Pnueli, A.: The temporal logic of programs. In: *FOCS*, pp. 46–57. IEEE Computer Society Press, Los Alamitos (1977)
12. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. *J. ACM* 32(3), 733–749 (1985)
13. Walukiewicz, I.: Pushdown processes: Games and model-checking. *Inf. Comput.* 164(2), 234–263 (2001)

# Paper Retraction: On the Hardness of Embeddings Between Two Finite Metrics

Matthew Cary, Atri Rudra, and Ashish Sabharwal

Computer Science and Engineering,  
University of Washington, Seattle, WA 98195-2350, U.S.A

We regret to report that we have found an error in our paper “On the Hardness of Embeddings Between Two Finite Metrics,” which appeared in the Proceedings of the 32<sup>nd</sup> International Colloquium on Automata, Languages and Programming (ICALP), Lisboa, Portugal, July 2005.

In that paper, we sought to prove NP-hardness of approximating the minimum distortion when one finite metric is embedded into another. While the fairly involved reduction we presented was correct, the Directed Disjoint Cycle Cover (DDCC) problem we were reducing from has a polynomial-time algorithm, as noticed later by Jiri Sgall. Despite much effort, we have not been able to find an alternate reduction, and it remains open whether our claimed hardness of approximation results still hold.

# Author Index

- Adida, Ben 484  
Alon, Noga 352, 435, 789  
Arenas, Marcelo 888  
Atserias, Albert 279, 558  
Attrapadung, Nuttapong 496
- Bansal, Ajay 472  
Bansal, Nikhil 28  
Bar-Noy, Amotz 219  
Barceló, Pablo 888  
Beame, Paul 134  
Bellare, Mihir 399, 411  
Berger, André 90  
Bienvenu, Laurent 643  
Björklund, Henrik 850  
Bläser, Markus 801  
Blömer, Johannes 65  
Bodirsky, Manuel 546  
Boigelot, Bernard 813  
Bojańczyk, Mikołaj 850  
Bošnački, Dragan 158  
Bozzelli, Laura 925  
Brihaye, Thomas 825  
Brusten, Julien 813  
Bulatov, Andrei 279, 558  
Bürgisser, Peter 207
- Cai, Jin-Yi 631  
Caragiannis, Ioannis 447  
Cary, Matthew 949  
Chan, Ho-Leung 28  
Chandran, Nishanth 423  
Chazelle, Bernard 1  
Cheilaris, Panagiotis 219  
Chen, Hubie 546  
Christodoulou, George 40  
Chu, Matthew 728  
Coja-Oghlan, Amin 777, 789  
Colcombet, Thomas 901  
Cucker, Felipe 207
- Dalmau, Victor 279  
Danggård, Ivan 2  
David, Matei 134  
Dawar, Anuj 558, 913
- Dedic, Nenad 255  
Dell, Holger 801  
Demaine, Erik D. 146  
Dershowitz, Nachum 291  
Dorn, Frederic 15
- Elkin, Michael 716  
Elkind, Edith 158
- Fellows, Michael R. 340  
Fertin, Guillaume 340  
Fiat, Amos 583  
Fiore, Marcelo 607  
Flammini, Michele 447  
Fomin, Fedor V. 15, 352  
Fraigniaud, Pierre 231  
Franceschini, Gianni 533  
Furukawa, Jun 496
- Gavoille, Cyril 231  
Genest, Blaise 158  
Gimbert, Hugo 850  
Goubault-Larrecq, Jean 764  
Grigni, Michelangelo 90  
Grohe, Martin 363, 913  
Groth, Jens 423  
Grüber, Magdalena 363  
Guha, Sudipto 704  
Guo, Jiong 375  
Gupta, Ankur 521  
Gupta, Gopal 472  
Gutin, Gregory 352  
Gutner, Shai 435
- Hàn, Hiệp 789  
Hasuo, Ichiro 619  
He, Meng 509  
Henzinger, Thomas A. 825  
Hermelin, Danny 340  
Hon, Wing-Kai 521  
Hur, Chung-Kil 607
- Ilcinkas, David 231  
Ishai, Yuval 243  
Iwama, Kazuo 110

Jacobs, Bart 619  
 Jurdziński, Marcin 838

Kang, Mihyun 789  
 Kannan, Sampath 728  
 Kaplan, Haim 583  
 Kapron, Bruce 328  
 Kára, Jan 546  
 Karianto, Wong 875  
 Katriel, Irit 171  
 Kenyon-Mathieu, Claire 171  
 Kiayias, Aggelos 316  
 Kobayashi, Naoki 740  
 Kontogiannis, Spyros C. 595  
 Korman, Amos 102  
 Koutsoupias, Elias 40  
 Kovács, Annamária 40  
 Kreutzer, Stephan 571, 913  
 Krivelevich, Michael 352

La Torre, Salvatore 925, 937  
 Laird, James 667  
 Larose, Benoît 267  
 Lee, Jonathan K. 680  
 Levy, Meital 583  
 Libkin, Leonid 888  
 Löding, Christof 875  
 Lu, Chi-Jen 183  
 Lu, Pinyan 631  
 Lüttgen, Gerald 752

Malka, Lior 328  
 Malkin, Tal 243  
 Mallya, Ajay 472  
 McGregor, Andrew 704, 728  
 Merkle, Wolfgang 643  
 Miller, Gary L. 655  
 Mohassel, Payman 255  
 Montanaro, Ashley 122  
 Moran, Tal 303  
 Moscardelli, Luca 447  
 Motwani, Rajeev 53  
 Mozes, Shay 146  
 Munro, J. Ian 509  
 Muthukrishnan, S. 533

Naewe, Stefanie 65  
 Namprepre, Chanathip 411  
 Naor, Moni 303

Neven, Gregory 411  
 Niedermeier, Rolf 375  
 Nishimura, Harumichi 110

O'Donnell, Ryan 195  
 Oertzen, Timo von 546  
 Olonetsky, Svetlana 219, 583  
 Ostrovsky, Rafail 387  
 Otto, Martin 571

Palsberg, Jens 680  
 Panagiotou, Konstantinos 777  
 Pandey, Omkant 387  
 Panigrahy, Rina 53  
 Parlato, Gennaro 937  
 Pattinson, Dirk 459  
 Pelc, Andrzej 231  
 Peled, Doron 158  
 Pereira, Fernando Magno Quintão 680  
 Pettie, Seth 78  
 Phillips, Todd 655  
 Pitassi, Toniann 134  
 Prabhu, Vinayak S. 825  
 Pruhs, Kirk 28

Rao, S. Srinivasa 509  
 Raskin, Jean-François 825  
 Raymond, Rudy 110  
 Ristenpart, Thomas 399  
 Rödl, Vojtěch 789  
 Rossman, Benjamin 146  
 Rudra, Atri 949

Sabharwal, Ashish 949  
 Sahai, Amit 387, 423  
 Saurabh, Saket 352  
 Schacht, Mathias 789  
 Schmitz, Sylvain 692  
 Schneider, Fred B. 12  
 Schröder, Lutz 459  
 Schweikardt, Nicole 571, 913  
 Segev, Gil 303  
 Shah, Rahul 521  
 Sheehy, Donald 655  
 Simon, Luke 472  
 Smorodinsky, Shakhar 219  
 Spirakis, Paul G. 595  
 Srinivasan, Venkatesh 328  
 Steger, Angelika 777



- Strauss, Martin J. 243  
Suto, Takashi 740
- Tesson, Pascal 267  
Thilikos, Dimitrios M. 15  
Trivedi, Ashutosh 838  
Tsai, Shi-Chun 183  
Tzameret, Iddo 291
- Upfal, Eli 171  
Uustalu, Tarmo 619
- Vialette, Stéphane 340  
Vitter, Jeffrey Scott 521  
Vogler, Walter 752
- Weimann, Oren 146  
Wikström, Douglas 484  
Wimmer, Karl 195  
Winter, Andreas 122  
Woelfel, Philipp 134  
Wright, Rebecca N. 243  
Wu, Hsin-Lung 183
- Xu, Ying 53
- Yamashita, Shigeru 110
- Zhou, Hong-Sheng 316  
Zielonka Wiesław 850