Steffen Hölldobler
Carsten Lutz
Heinrich Wansing (Eds.)

LNAI 5293

Logics in Artificial Intelligence

11th European Conference, JELIA 2008
Dresden, Germany, September/October 2008
Proceedings

Lecture Notes in Artificial Intelligence 5293

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Steffen Hölldobler Carsten Lutz
Heinrich Wansing (Eds.)

Logics in Artificial Intelligence

11th European Conference, JELIA 2008
Dresden, Germany, September 28 - October 1, 2008
Proceedings

Springer

Volume Editors

Steffen Hölldobler
Dresden University of Technology
Artificial Intelligence Institute
Dresden, Germany
E-mail: sh@iccl.tu-dresden.de

Carsten Lutz
Dresden University of Technology
Institute for Theoretical Computer Science
Dresden, Germany
E-mail: lutz@tcs.inf.tu-dresden.de

Heinrich Wansing
Dresden University of Technology
Institute of Philosophy
Dresden, Germany
E-mail: heinrich.wansing@tu-dresden.de

Library of Congress Control Number: 2008935450

CR Subject Classification (1998): I.2, F.4.1, D.1.6

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-87802-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-87802-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12530502 06/3180 5 4 3 2 1 0

Preface

This volume contains the papers selected for presentation at the 11th European Conference on Logics in Artificial Intelligence (or Journées Européennes sur la Logique en Intelligence Artificielle, JELIA), which was held from September 28 to October 1, 2008, at the Technische Universität Dresden, Germany.

In total, 98 research papers were submitted by researchers from 24 countries. Each submission was reviewed by at least three expert reviewers. The final decisions on the papers were taken during an electronic Program Committee meeting held on the Internet. The Program Committee accepted 32 research papers for presentation at the conference. This includes two summaries of master theses that won the best thesis award of the European Master's Programme in Computational Logic (EMCL) in 2006 and 2007, respectively: the paper by Magdalena Ortiz on "Extending CARIN to the Description Logics of the SH Family" and the one by Novak Novakovic on "Proof-Theoretic Approach to Deciding Subsumption and Computing Least Common Subsumer in EL w.r.t. Hybrid TBoxes."

The program also included three invited lectures by Sergei Artemov, Ruth Byrne, and Jérôme Lang. The lecture by Ruth Byrne was given jointly to JELIA 2008 and to the 9. Fachtagung der Gesellschaft für Kognitionswissenschaft (9th Symposium of the German Cognitive Science Society), which was held in parallel at the Technische Universität Dresden. Also colocated with JELIA 2008 were the 9th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA-IX) and the 22nd Workshop on (Constraint) Logic Programming (WLP 2008).

Many people contributed to making JELIA 2008 a success. We thank the authors of the submitted papers, which were of very high quality and covered a broad range of topics including belief revision, description logics, non-monotonic reasoning, multi-agent systems, probabilistic logic, and temporal logic. We are grateful to the Program Committee and the referees for the considerable efforts that they invested in reviewing and selecting the papers. Andrei Voronkov's EasyChair conference software was of tremendous help during all phases of preparation of the conference and proceedings. We thank Bertram Fronhöfer, Axel Großmann, and Julia Koppenhagen for their continuous support and help with the local organization. A final thank you goes to Moritz Weeger for creating and maintaining the JELIA 2008 webpages as well as to Julia Schwung and Philipp Böhnke for designing the JELIA 2008 poster.

July 2008

Steffen Hölldobler
Carsten Lutz
Heinrich Wansing

Organization

Program Chairs

Steffen Hölldobler
Carsten Lutz
Heinrich Wansing

Program Committee

Jose Julio Alferes
Alessandro Artale
Chitta Baral
Peter Baumgartner
Alex Bochman
Gerhard Brewka
Walter Carnielli
James P. Delgrande
Jürgen Dix
Roy Dyckhoff
Thomas Eiter
Michael Fisher
Ulrich Furbach
Michael Gelfond
Rajeev Gore
Sven Ove Hansson
James Harland
Andreas Herzig
Pascal Hitzler
Tomi Janhunen
Peter Jonsson
Lluís Godo Lacasa
Nicola Leone
Vladimir Lifschitz
John-Jules Meyer
Angelo Montanari
Bernhard Nebel
Ilkka Niemelä
David Pearce
Luis Moniz Pereira
Henry Prakken
Francesca Rossi
Torsten Schaub
Renate Schmidt
Lim Yohanes Stefanus
Umberto Straccia
Michael Thielscher
Francesca Toni
Mirosław Truszczyński
Andrei Voronkov
Toby Walsh
Michael Zakharyaschev
Ruy de Queiroz
Wiebe van der Hoek

Local Arrangements

Axel Großmann
Bertram Fronhöfer
Julia Koppenhagen

External Reviewers

Teresa Alsinet
Sergey Babenyshev
Federico Banti
Tristan Behrens
Marta Bilkova
Jürgen Bock
Kai Brünnler
Nils Bulling
Willem Conradie
Carlos Damásio
Agostino Dovier
Conrad Drescher
Steve Dworschak
Uwe Egly
Wolfgang Faber
Tim French
Bertram Fronhöfer
Benoit Gaudou
Martin Gebser
Rosella Gennari
Ingo Glöckner
Rodolfo Gómez
Giovanni Grasso
Stephan Grimm
Aaron Hunter
Ullrich Hustadt
Maria Keet
Matthias Knorr
E. Komendantskaya
Vladimir Komendantsky

Roman Kontchakov
João Leite
Alexei Lisitsa
Iain Little
Gallucci Lorenzo
Emiliano Lorini
Marco Maratea
Markus Maron
Alessandra Mileo
Boris Motik
Alexander Nittka
Peter Novak
Claudia Obermaier
Emilia Oikarinen
Magdalena Ortiz
Gabriele Puppis
Maurice Pagnucco
Linda Postniece
Guilin Qi
Marco Ragni
Hilverd Reker
Jochen Renz
Francesco Ricca
Jussi Rintanen
Vladislav Ryzhikov
Gabriele Röger
Pietro Sala
Ansgar Scherp
Stephan Schiffel
Thomas Schneider
Guido Sciavicco
Anthony Seda
Inanç Seylan
Mantas Šimkus
Volker Sorge
Giorgio Terracina
Sven Thiele
Dmitry Tishkovsky
Yiorgos Trimponias
Nicolas Troquard
Tuvshintur Tserendorj
Alexandra Uitdenbogerd
Joost Vennekens
Jürgen Villadsen
Nicola Vitacolonna
Dirk Walther
Richard Watson
Volker Weber
Gregory Wheeler
Florian Widmann
Nic Wilson
Stefan Wölfl
Stefan Woltran
Mike Wooldridge
Guillaume Aucher
Hans van Ditmarsch

Table of Contents

Invited Talks

Regular Papers

Justification Logic

Sergei Artemov

CUNY Graduate Center,
365 Fifth Ave., New York City, NY 10016, USA
SArtemov@gc.cuny.edu

Justification Logic offers a new approach to a theory of knowledge, belief, and evidence, which possesses the potential to have significant impact on applications. The celebrated account of **knowledge** as **justified true belief**, which is attributed to Plato, has long been a focus of epistemic studies (cf. [10,15,18,26,30,32] and many others).

About a half-century ago, the notions of knowledge and belief acquired formalization by means of modal logic ([22,34]). Within this approach, the following analysis is adopted:

F is known

is interpreted as

F holds in all possible situations.

The resulting *epistemic logic* has been remarkably successful in terms of developing a rich mathematical theory and applications (cf. [13,28], and other sources). However, it misses the mark in some situations:

> *What if F holds at all possible worlds, e.g., a mathematical truth, say $P \neq NP$, but the agent is simply not aware of the fact due to lack of evidence, proof, justification, etc.?*

The notion of justification, an essential element of epistemic studies, was conspicuously absent in epistemic logic, which led to well-known deficiencies inherent in modal logics of knowledge. This is displayed most prominently in the *Logical Omniscience* defect of the modal logic of knowledge (cf. [11,12,23,29,31]). According to a basic principle of epistemic modal logic,

$$\Box(F \to G) \to (\Box F \to \Box G),$$

the agent is supposed to "know" all logical consequences of his/her assumptions. In particular, *each agent who knows the rules of Chess should know whether there is a winning strategy for White.* Furthermore, *if an agent knows a product of two (very large) primes, then the agent should know each of the primes*, etc.

Within the provability domain, the absence of an adequate description of the logic of justifications (here mathematical proofs) remained an impediment to both the formalizing of Brouwer-Heyting-Kolmorogorov semantics for intuitionistic logic ([16,17,21,24]) and providing an intended semantics for Gödel's provability logic S4 ([2,3,6,16,17,33]). This lack of a justification component has, perhaps, contributed to a certain gap between epistemic logic and mainstream epistemology ([19,20]).

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 1–4, 2008.

We describe a general logical framework, Justification Logic, for reasoning about epistemic justification ([1,3,4,5,7,8,9,14,25] and others). Justification Logic is based on classical propositional logic augmented by justification assertions $t{:}F$ that read

t is a justification for F.

Justification Logic absorbs basic principles originating from both mainstream epistemology and the mathematical theory of proofs.

As a case study, we formalize Gettier examples ([15]) in Justification Logic and reveal hidden assumptions and redundancies in Gettier reasoning. Furthermore, we formalize Kripke's well-known 'red barn example' [27] and offer a resolution of the corresponding paradox. The latter provides a clear example of a problem which lies outside the scope of the traditional epistemic modal logic but can be handled naturally in Justification Logic.

We state a general Correspondence Theorem showing that behind each epistemic modal logic, there is a robust system of justifications.

Justification Logic extends the logic of knowledge in three major ways.

First, it adds a long-anticipated mathematical notion of justification, making the logic more expressive. We now have the capacity to reason about justifications, simple and compound. We can compare different pieces of evidence pertaining to the same fact. We can measure the complexity of justifications, thus connecting the logic of knowledge to a rich complexity theory, etc.

Second, justification logic furnishes a new, evidence-based foundation for the logic of knowledge, according to which

F is known

is interpreted as

F has an adequate justification.

Third, justification logic provides a novel, evidence-based mechanism of truth tracking which can be a valuable tool for extracting robust justifications from a larger body of justifications which are not necessarily reliable.

Knowledge, belief, and evidence are fundamental concepts whose significance spans many areas of human activity: artificial intelligence and computer science, mathematics, economics and game theory, cryptography, philosophy, and other disciplines. Justification Logic promises significant impact on the aforementioned areas. The capacity to keep track of pieces of evidence, compare them, and select those that are appropriate is a valuable new tool.

References

1. Artemov, S.: Operational modal logic. Technical Report MSI 95-29, Cornell University (1995)
2. Artemov, S.: Understanding constructive semantics. Spinoza Lecture for European Association for Logic, Language and Information, Utrecht (1999)
3. Artemov, S.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)

4. Artemov, S.: Justified common knowledge. Theoretical Computer Science 357(1-3), 4–22 (2006)
5. Artemov, S.: Justification logic. Technical Report TR-2007019, CUNY Ph.D. Program in Computer Science (2007)
6. Artemov, S., Beklemishev, L.: Provability logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn., pp. 189–360. Springer, Heidelberg (2005)
7. Artemov, S., Kuznets, R.: Logical omniscience via proof complexity. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 135–149. Springer, Heidelberg (2006)
8. Artemov, S., Nogina, E.: Introducing justification into epistemic logic. J. of Logic and Computation 15(6), 1059–1073 (2005)
9. Brezhnev, V., Kuznets, R.: Making knowledge explicit: How hard it is. Theoretical Computer Science 357(1-3), 23–34 (2006)
10. Dretske, F.: Conclusive reasons. Australasian Journal of Philosophy 49, 1–22 (1971)
11. Fagin, R., Halpern, J.: Belief, awareness, and limited reasoning: Preliminary report. In: Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI 1985), pp. 491–501 (1985)
12. Fagin, R., Halpern, J.: Belief, awareness, and limited reasoning. Artificial Intelligence 34(1), 39–76 (1988)
13. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about knowledge. MIT Press, Cambridge (1995)
14. Fitting, M.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132(1), 1–25 (2005)
15. Gettier, E.: Is justified true belief knowledge? Analysis 23, 121–123 (1963)
16. Gödel, K.: Eine Interpretation des intuitionistischen Aussagenkalkuls. Ergebnisse Math. Kolloq. 4, 39–40 (1933); English translation in: Feferman, S., et al. (eds) Kurt Gödel Collected Works, Vol. I, pp. 301–303. Oxford University Press, Oxford (1986)
17. Gödel, K.: Vortrag bei Zilsel. In: English translation, Feferman, S., et al. (eds.) Kurt Gödel Collected Works, vol. III, pp. 86–113. Oxford University Press, Oxford (1995)
18. Goldman, A.: A causal theory of knowing. The Journal of Philosophy 64, 335–372 (1967)
19. Hendricks, V.F.: Active agents. Journal of Logic, Language and Information 12(4), 469–495 (2003)
20. Hendricks, V.F.: Mainstream and formal epistemology. Cambridge University Press, New York (2005)
21. Heyting, A.: Mathematische Grundlagenforschung. Intuitionismus. Beweistheorie. Springer, Berlin (1934)
22. Hintikka, J.: Knowledge and belief. Cornell University Press, Ithaca (1962)
23. Hintikka, J.: Impossible possible worlds vindicated. Journal of Philosophical Logic 4, 475–484 (1975)
24. Kolmogoroff, A.: Zur Deutung der intuitionistischen logik. Math. Z. 35, 58–65 (1932) (in German); English transl.: Selected works of Kolmogorov, A.N., vol. I, Mathematics and Mechanics, Tikhomirov, V.M. (ed.). Kluwer, Dordrecht, pp. 151–158 (1991)
25. Kuznets, R.: On the complexity of explicit modal logics. In: Clote, P.G., Schwichtenberg, H. (eds.) CSL 2000. LNCS, vol. 1862, pp. 371–383. Springer, Heidelberg (2000)
26. Lehrer, K., Paxson, T.: Knowledge: undefeated justified true belief. The Journal of Philosophy 66, 1–22 (1969)

27. Luper, S.: The epistemic closure principle. Stanford Encyclopedia of Philosophy (2005)
28. Meyer, J.-J.Ch., van der Hoek, W.: Epistemic logic for AI and computer science. Cambridge (1995)
29. Moses, Y.: Resource-bounded knowledge. In: Vardi, M. (ed.) Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge, Pacific Grove, California, March 7–9, 1988, pp. 261–276. Morgan Kaufmann Pbl, San Francisco (1988)
30. Nozick, R.: Philosophical explanations. Harvard University Press (1981)
31. Parikh, R.: Knowledge and the problem of logical omniscience. In: Ras, Z., Zemankova, M. (eds.) ISMIS 1987 (International Symposium on Methodology for Intellectual Systems), pp. 432–439. North-Holland, Amsterdam (1987)
32. Stalnaker, R.C.: Knowledge, belief and counterfactual reasoning in games. Economics and Philosophy 12, 133–163 (1996)
33. van Dalen, D.: Intuitionistic logic. In: Gabbay, D., Guenther, F. (eds.) Handbook of Philosophical Logic, Reidel, vol. 3, pp. 225–340 (1986)
34. von Wright, G.H.: An essay in modal logic. North-Holland, Amsterdam (1951)

Voting in Combinatorial Domains: What Logic and AI Have to Say

(Extended Abstract)

Jérôme Lang

IRIT - Université Paul Sabatier and CNRS
31062 Toulouse Cedex, France
lang@irit.fr

This talk is half-way between a survey and a report on ongoing research, most of which is joint work with Vincent Conitzer and Lirong Xia. The first part of the talk owes a lot to two survey papers co-authored with Yann Chevaleyre, Ulle Endriss and Nicolas Maudet [5, 6].

Social Choice Theory is the subfield of economics that aims at designing and evaluating methods for collective decision making [1]. Most works in the field focus on normative questions: the problem is generally considered to be solved when the existence (or the non-existence) of a procedure meeting some requirements has been shown; how hard it is to compute this procedure, and how it should be computed, have deserved much less attention in the Social Choice community. This is where Computer Science, and more specifically Artificial Intelligence, come into play, giving birth to a new interdisciplinary research field called *Computational Social Choice* (see [5] for an introduction to the field). *The first part of the talk will consist of a brief introduction to computational social choice.*

One of the hot topics in computational social choice is *voting on a set of alternatives that has a combinatorial structure*: in other words, the voters have to make a common decision on several possibly related issues. For instance, the inhabitants of some local community may have to make a joint decision over several related issues of local interest, such as deciding whether some new public facility such as a swimming pool or a tennis court should be built. Such elections are called *multiple referenda* [4]. Some of the voters may have preferential dependencies, for instance, they may prefer the tennis court to be built only if the swimming pool is not. Another example is when the members of an association have to elect a steering committee, composed of a president, a vice-president and a treasurer (see for instance [2]). Again, voters typically have preferential dependencies.

As soon as voters have preferential dependencies between issues, it is generally a bad idea to decompose the problem into a set of p smaller problems, each one bearing on a single issue. Doing so typically lead to "multiple election paradoxes". Such paradoxes have been studied by a number of authors [4, 2, 8]. Consider the following example. A joint decision has to be made about whether or not to build a new swimming pool (S or $\bar{S}$) and a new tennis court (T or $\bar{T}$). Assume that the preferences of voters 1 and 2 are $S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T} \succ ST$, those of voters 3 and 4 are $\bar{S}T \succ S\bar{T} \succ \bar{S}\bar{T} \succ ST$, and those of voter 5 are $ST \succ S\bar{T} \succ \bar{S}T \succ \bar{S}\bar{T}$. The first problem is that voters 1

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 5–7, 2008.

to 4 will feel ill at ease when asked to report their projected preference on $\{S, \bar{S}\}$ and $\{T, \bar{T}\}$. Only voter 5 knows that whatever the other voters' preferences about $\{S, \bar{S}\}$ (resp. $\{T, \bar{T}\}$), she can vote for T (resp. S) without any risk of experiencing regret. Experimental studies suggest that most voters tend to report their preferences optimistically in such situations; for instance, voters 1–2 would likely report a preference for S over $\bar{S}$. The second problem (the paradox itself) is that under this assumption that voters report optimistic preferences, the outcome will be $S\bar{T}$, which is *the worst outcome for all but one voter*. A difficult question is how to design a method for voting on related issues that avoids such paradoxes.

We can list five ways of proceeding, each of which has its own pitfalls: (1) ask voters to report their entire preference relation explicitly on the set of alternatives, and then apply a fixed voting rule; (2) ask voters to report only a small part of their preference relation (for instance, their k preferred outcomes, where k is a small number) and apply a voting rule that needs this information only; (3) limit the number of possible combinations that voters may vote for, as advocated in [4]; (4) ask each voter to express her preferences as an input in some fixed compact representation language, and then apply a fixed voting rule to the profile consisting of the preference relations induced by the voters' inputs (see [9]); and finally, (5) impose a domain restriction allowing the voters to vote separately on each issue, either simultaneously or sequentially. Especially, if the voters' preferences are *separable* (which means that each voter's preference on the values of an issue is independent from the outcome on other issues), then this approach is reasonably safe, as shown in [8].

The second part of the talk will present the general problem of voting on combinatorial domains, as well as its paradoxes, and will briefly discuss the pros and cons of these five possible classes of solutions.

Then we focus on the last class of solutions, which seems to be the most promising. Its main drawback is that separability is a very demanding assumption, and is unlikely to be met in practice. Several recent papers [10,13,14,12] impose a much weaker domain restriction under which sequential voting can be applied "safely": each time a voter is asked to report her preferences on an issue, these preferences do not depend on the values of the issues that have not been decided yet. Formally, this can be expressed as the following condition: there is a linear order $\mathcal{O} = \mathbf{X}_1 > \ldots > \mathbf{X}_p$ on the set of issues such that the preferences of each voter on $\mathbf{X}_j$ are preferentially independent from $\mathbf{X}_{j+1}, \ldots, \mathbf{X}_p$ given $\mathbf{X}_1, \ldots, \mathbf{X}_{j-1}$. If this property is satisfied, then *sequential voting rules* can be defined in the following way. Let $r_1, \ldots, r_p$ be voting rules on the domains of $\mathbf{X}_1, \ldots, \mathbf{X}_p$ respectively. The *sequential composition* of $r_1, \ldots, r_p$ is defined by applying the following protocol repeatedly for $i = 1, \ldots, n$: elicit voters' preferences on the domain of $\mathbf{X}_i$; apply r_i to these local preferences; broadcast the outcome to the voters. Clearly, in order to compute the outcome of these sequential voting rules we do not need to know the voters' full preference relations: it suffices for each voter to express a CP-net [3], with the condition that the dependency graph of the CP-nets is *acyclic* and *common to all voters*.

Even if the domain restriction imposed by sequential voting it is much weaker than separability, it still eliminates most of the profiles. This restriction is weakened in [14]: profiles must only be compatible with *some* common order, not specified in the

definition of the sequential voting rule. [12] goes much further and give a generalization sequential voting rules that does not require any domain restriction. However, the gain in generality comes with a complexity gap.

The third part of the talk will present sequential voting rules and these latter two generalizations.

Coming back to the problem that voters encounter when asked to report their projected preferences on single issues (in the example above, between $\{S, \bar{S}\}$ and $\{T, \bar{T}\}$): when expressing such preferences, voters have to lift their preferences from the level of alternatives to the level of *sets of alternatives*, or equivalently, to the level of propositional formulas (with issues as propositional variables). This suggests the existence of a strong connection between voting on combinatorial domains and *preference logics* [11,7]. *This will be the topic of the last part of the talk.*

References

1. Arrow, K.J., Sen, A.K., Suzumura, K.: Handbook of Social Choice and Welfare. North-Holland, Amsterdam (2002)
2. Benoit, J.-P., Kornhauser, L.: On the assumption of separable assembly preferences. Social Choice and Welfare 16(3), 429–439 (1999)
3. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. Journal of Artificial Intelligence Research 21, 135–191 (2004)
4. Brams, S., Kilgour, D., Zwicker, W.: The paradox of multiple elections. Social Choice and Welfare 15(2), 211–236 (1998)
5. Chevaleyre, Y., Endriss, U., Lang, J., Maudet, N.: A short introduction to computational social choice. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362. Springer, Heidelberg (2007)
6. Chevaleyre, Y., Endriss, U., Lang, J., Maudet, N.: Preference handling in combinatorial domains: From AI to social choice. AI Magazine (to appear, 2008)
7. Hansson, S.O.: Preference logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, pp. 319–393. Kluwer, Dordrecht (2001)
8. Lacy, D., Niou, E.: A problem with referenda. Journal of Theoretical Politics 12(1), 5–31 (2000)
9. Lang, J.: Logical preference representation and combinatorial vote. Annals of Mathematics and Artificial Intelligence 42(1–3), 37–71 (2004)
10. Lang, J.: Voting and aggregation on combinatorial domains with structured preferences. In: Proc. 20th Joint International Conference on Artificial Intelligence (IJCAI 2007) (2007)
11. von Wright, H.H.: The logic of preference. Edinburgh University Press (1963)
12. Xia, L., Conitzer, V., Lang, J.: Voting on multiattribute domains with cyclic preferential dependencies – a new family of voting rules. In: Proceedings of AAAI 2008 (2008)
13. Xia, L., Lang, J., Ying, M.: Sequential voting and multiple election paradoxes. In: Proc. 11th Conference on the Theoretical Aspects of Rationality and Knowledge (TARK 2007) (2007)
14. Xia, L., Lang, J., Ying, M.: Strongly decomposable voting rules on multiattribute domains. In: Proceedings of AAAI 2007 (2007)

Strongly Equivalent Temporal Logic Programs*

Felicidad Aguado, Pedro Cabalar, Gilberto Pérez, and Concepción Vidal

Dept. Computación,
Corunna University, Spain
{aguado,cabalar,gperez,eicovima}@udc.es

Abstract. This paper analyses the idea of strong equivalence for transition systems represented as logic programs under the Answer Set Programming (ASP) paradigm. To check strong equivalence, we use a linear temporal extension of Equilibrium Logic (a logical characterisation of ASP) and its monotonic basis, the intermediate logic of Here-and-There (HT). Trivially, equivalence in this temporal extension of HT provides a sufficient condition for temporal strong equivalence and, as we show in the paper, it can be transformed into a provability test into the standard Linear Temporal Logic (LTL), something that can be automatically checked using any of the LTL available provers. The paper shows an example of the potential utility of this method by detecting some redundant rules in a simple actions reasoning scenario.

1 Introduction

The paradigm of *Answer set programming* (ASP) [1,2] (based on the *stable models* semantics [3]) constitutes one of the most successful examples of logical nonmonotonic formalisms applied to Knowledge Representation [4,5] in Artificial Intelligence. Probably, the reasons for this success are both related to its powerful representational features and, at the same time, to the availability of an increasing number of efficient ASP solvers (see [6]) that allow its application to many real scenarios. Concerning the formalism properties, ASP is characterised by providing nonmonotonic reasoning with a rich and flexible syntax, initially born from logic programming, but continuously extended thereafter along the research history in the area, without overlooking its original semantic simplicity. An important breakthrough in this sense has been the logical characterisation of ASP in terms of *Equilibrium Logic* [7] that has opened, for instance, the study of strong equivalence [8] (the main topic of this paper) and has recently allowed the extension of the stable models semantics for arbitrary first order theories [9,10].

As for the practical applications of ASP, perhaps one of the most outstanding and frequent uses has been the representation and automated reasoning for action domains, solving typical problems like prediction, explanation, planning

* This research is partially supported by Spanish Ministry MEC, research project TIN-15455-C03-02.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 8–20, 2008.

or diagnostics. Default negation plays here a crucial role, as it allows representing the rule of *inertia* (that can be stated as "a fluent remains unchanged by default") and avoid in this way the frame problem [11]. ASP can also be naturally used for solving other typical representational problems in Reasoning about Actions and Change, and is in fact the basis for a family[1] of high level action languages [14]. The use of ASP solvers for action domains, however, has some limitations, as partly explained by the complexity class that these solvers allow to capture (Σ_2^P in the most general case). In practice, this means for instance that, when solving a planning problem (which lies in PSPACE-completeness) we must fix an *a priori* plan length so that a ground logic program can be eventually generated. The search for a minimal plan consists then in gradually incrementing this plan length until a stable model is found. A first obvious drawback of this approach is that it is not possible to establish when a given planning problem has *no solution* of any length at all. A second and more elaborated problem is that it is impossible to establish when two descriptions of the same transition system are *strongly equivalent*, i.e., when they will behave in the same way regardless any additional rules we include and any path length we consider.

At a first sight, it could be thought that this problem for checking strong equivalence of temporal scenarios is only due to the restriction to ground programs. For instance, there already exists a tool (`SE-TEST` [15]) that allows checking strong equivalence of logic programs with variables, without requiring their previous grounding. However, temporal domains require something else than variables: in order to represent transition rules, we must (at least) be able to refer to a situation index variable I and its successor situation $I+1$. This forces us either to deal with integer numbers, or at least, with a Peano like representation of the form $s(I)$ with a function symbol s to represent the successor. Unfortunately, `SE-TEST` is exclusively thought for Datalog programs, i.e. functions or unbounded integers are not allowed.

As we mentioned above, the idea of *strong equivalence* was introduced in [8], where the following question was considered: when can we safely replace a piece of knowledge representation by an "equivalent" one independently of the context? Formally, we say that two logic programs Π_1 and Π_2 are *strongly equivalent* when, for any arbitrary logic program Π, both $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ have the same stable models. Note that, for a monotonic logic, this property trivially collapses to regular equivalence (i.e., coincidende of sets of models) of Π_1 and Π_2. However, when a nonmonotonic entailment is involved, the addition of a set of rules Π may have different effects on the sets of stable models of Π_1 and Π_2, so that strong equivalence is indeed a *stronger* property than regular equivalence. In [8] it was shown that two logic programs are strongly equivalent (under the stable models semantics) if and only if they are equivalent under the intermediate logic of *Here-and-There* [18], the monotonic basis of Equilibrium Logic.

[1] To be precise, part of this family relies on the formalism of Causal Theories [12] but, as shown in [13], this formalism can be reduced to ASP as well.

In this paper we consider the study of strong equivalence for logic programs that represent transition systems. To this aim, we revisit a temporal extension of Equilibrium Logic proposed in [19] which consists in the inclusion of modal operators as those handled in Linear Temporal Logic (LTL) [20,21]. This extension, called *Temporal Equilibrium Logic*, immediately provides us with a sufficient condition for strong equivalence of temporal logic programs: we can simply check regular equivalence in its monotonic basis, a logic we called *Temporal Here-and-There* (THT) [19]. The main contribution of the paper is the automation of this test for strong equivalence so that, using a similar translation to those presented in [22,23,10], we transform a THT formula into LTL and use an LTL prover afterwards – in particular, we ran our experiments on the *Logics Workbench* [24].

The paper is organised as follows. In the next section, we introduce a simple motivating example, extracted from the Reasoning about Actions literature, to show the kind of problems we are interested in, proposing a pair of strong equivalence tests in this domain. In Section 3 we revisit the syntax and semantics of *Temporal Here-and-There* (THT) and we propose a models selection criterion to define the nonmonotonic *Temporal Equilibrium Logic* (TEL). Section 4 presents the translation from THT into LTL, whereas Section 5 applies this translation to answer the questions proposed in Section 3. Finally, Section 6 contains the conclusions and future work.

2 A Simple Motivating Example

Consider the following simple and well-known scenario [25] from Reasoning about Actions literature.

Example 1. An electric circuit consists of a battery, two switches and a light bulb. The switches are serially connected, as shown in Figure 2. The system state is expressed in terms of three propositional fluents sw_1, sw_2 and $light$, whose negations are represented with a bar on top of each fluent symbol. The state of each switch sw_i can be alternated by performing a corresponding action $toggle_i$. □

For simplicity, we assume that we do not handle concurrent actions. A possible representation of this scenario as an ASP logic program, Π_1, is shown below:

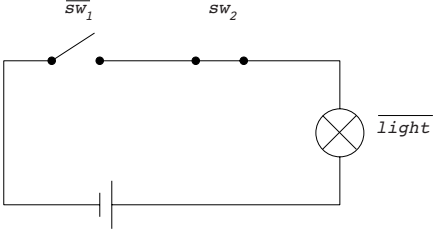


Fig. 1. A simple electric circuit

$$sw_j(I+1) \leftarrow toggle_j(I), \overline{sw}_j(I) \quad (1)$$
$$\overline{sw}_j(I+1) \leftarrow toggle_j(I), sw_j(I) \quad (2)$$
$$\overline{light}(I+1) \leftarrow toggle_j(I), sw_j(I) \quad (3)$$
$$light(I+1) \leftarrow toggle_1(I), \overline{sw}_1(I), sw_2(I) \quad (4)$$
$$light(I+1) \leftarrow toggle_2(I), \overline{sw}_2(I), sw_1(I) \quad (5)$$
$$\bot \leftarrow toggle_1(I), toggle_2(I) \quad (6)$$
$$f(I+1) \leftarrow f(I), not\ \overline{f}(I+1) \quad (7)$$
$$\overline{f}(I+1) \leftarrow \overline{f}(I), not\ f(I+1) \quad (8)$$
$$\bot \leftarrow f(I), \overline{f}(I) \quad (9)$$

where $j \in \{1,2\}$ and $f \in \{sw_1, sw_2, light\}$, and we assume that each symbol like $\overline{f}$ is actually treated as a new predicate. Variable $I = 0 \ldots n-1$ represents an integer index for each situation in the temporal narrative. Rules (1)-(5) are the *effect axioms*, capturing all the direct effects of actions. Rule (6) just avoids performing concurrent actions. Rules (7) and (8) represent the inertia default for each fluent f. Finally, (9) expresses that each fluent f and its explicit negation $\overline{f}$ cannot be simultaneously true[2].

As explained in the introduction, a planning problem would include additional facts (representing the initial state) and rules for generating actions and expressing a plan goal. But the most important additional step is deciding a finite limit n for variable I, so that the above program can be grounded.

Assume now that we want to modify Π_1 after noticing that the truth value of *light* is actually determined by the value of the two switches, becoming in this way an indirect effect or *ramification*[3]. In other words, we consider the addition of rules:

$$light(I) \leftarrow sw_1(I), sw_2(I) \quad (10)$$
$$\overline{light}(I) \leftarrow \overline{sw_1}(I) \quad (11)$$
$$\overline{light}(I) \leftarrow \overline{sw_2}(I) \quad (12)$$

If we call $\Pi_2 = \Pi_1 \cup \{(10) - (12)\}$, we may reasonably propose the following questions:

(Q1) Can we safely remove now from Π_2 the effect axioms for fluent *light* (3)-(5)?

(Q2) Assume we removed (3)-(5). Since *light* is "defined" now in terms of sw_1 and sw_2, can we safely remove from Π_2 the inertia rules for *light*?

In order to answer these questions, the current tools for testing strong equivalence for ground programs [26,27] cannot provide successful answers in an automated way, as they *need* a numeric value for the path length n to be previously fixed.

[2] Although perhaps not needed in this particular case, this axiom must be part of any general translation of an action theory. It is not difficult to see how the above scenario can be easily modified to introduce new contradictory effects that would not be detected in absence of (9).

[3] In fact, this example were used in [25] to illustrate possible representational problems for a suitable treatment of action ramifications.

3 Linear Temporal Here-and-There (THT)

We proceed now to recall the main definitions of the temporal extension of Equilibrium Logic, beginning with its monotonic basis. The logic of *Linear Temporal Here-and-There* (THT) was defined as follows. We start from a finite set of atoms Σ called the *propositional signature*. A (temporal) *formula* is defined as any combination of the atoms in Σ with the classical connectives $\wedge, \vee, \rightarrow, \bot$, the (unary) temporal operators $\Box$ (to be read as "always" or "from now on"), $\Diamond$ ("eventually") , $\bigcirc$ ("next") and the (binary) temporal operators $\mathcal{U}$ ("until"), $\mathcal{W}$ ("weak until") and $\mathcal{B}$ ("before"). Negation is defined as $\neg\varphi \stackrel{def}{=} \varphi \rightarrow \bot$ whereas $\top \stackrel{def}{=} \neg\bot$. As usual, $\varphi \leftrightarrow \psi$ stands for $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$. We also allow the abbreviation $\bigcirc^i\varphi \stackrel{def}{=} \bigcirc(\bigcirc^{i-1}\varphi)$ for $i > 0$ and $\bigcirc^0\varphi \stackrel{def}{=} \varphi$.

A (temporal) *interpretation* M is an infinite sequence of pairs $\langle H_i, T_i\rangle$ with $i = 0, 1, 2, \ldots$ where $H_i \subseteq T_i$ are sets of atoms. From an ASP point of view and informally speaking, atoms in T_i would play the role of an interpretation (that depends on a time parameter i) we would use for building a program reduct, whereas atoms in H_i will be minimised and later required to coincide with T_i. For simplicity, given a temporal interpretation, we write $\overline{H}$ (resp. $\overline{T}$) to denote the sequence of pair components $H_0, H_1, \ldots$ (resp. $T_0, T_1, \ldots$). Using this notation, we will sometimes abbreviate the interpretation as $M = \langle \overline{H}, \overline{T}\rangle$. An interpretation $M = \langle \overline{H}, \overline{T}\rangle$ is said to be total when $\overline{H} = \overline{T}$. We say that an interpretation $M = \langle \overline{H}, \overline{T}\rangle$ satisfies a formula φ at some sequence index i, written $M, i \models \varphi$, when any of the following hold:

1. $M, i \models p$ if $p \in H_i$, for any atom p.
2. $M, i \models \varphi \wedge \psi$ if $M, i \models \varphi$ and $M, i \models \psi$.
3. $M, i \models \varphi \vee \psi$ if $M, i \models \varphi$ or $M, i \models \psi$.
4. $\langle \overline{H}, \overline{T}\rangle, i \models \varphi \rightarrow \psi$ if both:
   (a) $\langle \overline{H}, \overline{T}\rangle, i \not\models \varphi$ or $\langle \overline{H}, \overline{T}\rangle \models \psi$;
   (b) $\langle \overline{T}, \overline{T}\rangle, i \not\models \varphi$ or $\langle \overline{T}, \overline{T}\rangle \models \psi$.
5. $M, i \models \bigcirc\varphi$ if $M, i+1 \models \varphi$.
6. $M, i \models \Box\varphi$ if for all $j \geq i$, $M, j \models \varphi$.
7. $M, i \models \Diamond\varphi$ if there exists some $j \geq i$, $M, j \models \varphi$.
8. $M, i \models \varphi \,\mathcal{U}\, \psi$ if there exists $j \geq i$, $M, j \models \psi$ and $M, k \models \varphi$ for all k such that $i \leq k < j$.
9. $M, i \models \varphi \,\mathcal{W}\, \psi$ if either $M, i \models \varphi \,\mathcal{U}\, \psi$ or, for all $j \geq i$, $M, j \models \varphi$.
10. $M, i \models \varphi \,\mathcal{B}\, \psi$ if for all $j \geq i$, either $M, j \models \psi$ or there exists some k, $i \leq k < j$ such that $M, k \models \varphi$.

We assume that a finite sequence $\langle H_0, T_0\rangle \ldots \langle H_n, T_n\rangle$ with $n \geq 0$ is an abbreviation of the infinite sequence $\langle \overline{H}', \overline{T}'\rangle$ with $H'_i = H_i$, $T'_i = T_i$ for $i = 0, \ldots, n$ and $H'_i = H_n$, $T'_i = T_n$ for $i > n$.

The logic of THT is an orthogonal combination of the logic of HT and the (standard) linear temporal logic (LTL) [21]. When we restrict temporal interpretations to finite sequences of length 1, that is $\langle H_0, T_0\rangle$ and disregard temporal

operators, we obtain the logic of HT. On the other hand, if we restrict the semantics to total interpretations, $\langle \overline{T}, \overline{T} \rangle, i \models \varphi$ corresponds to satisfaction of formulas $\overline{T}, i \models \varphi$ in LTL.

A *theory* is any set of formulas. An interpretation M is a *model* of a theory Γ, written $M \models \Gamma$, if $M, 0 \models \alpha$, for each formula $\alpha \in \Gamma$. A formula φ is *valid* if $M, 0 \models \varphi$ for any M. The following are valid formulas in THT (and in LTL too):

$$\Diamond\varphi \leftrightarrow \top \,\mathcal{U}\, \varphi \tag{13}$$
$$\Box\varphi \leftrightarrow \bot \,\mathcal{B}\, \varphi \tag{14}$$
$$\varphi \,\mathcal{W}\, \psi \leftrightarrow \varphi \,\mathcal{U}\, \psi \vee \Box\varphi \tag{15}$$
$$\varphi \,\mathcal{U}\, \psi \leftrightarrow \varphi \,\mathcal{W}\, \psi \wedge \Diamond\psi \tag{16}$$
$$\neg(\varphi \,\mathcal{U}\, \psi) \leftrightarrow \neg\varphi \,\mathcal{B}\, \neg\psi \tag{17}$$
$$\neg(\varphi \,\mathcal{B}\, \psi) \leftrightarrow \neg\varphi \,\mathcal{U}\, \neg\psi \tag{18}$$
$$\neg\Box\varphi \leftrightarrow \Diamond\neg\varphi \tag{19}$$
$$\neg\Diamond\varphi \leftrightarrow \Box\neg\varphi \tag{20}$$
$$\bigcirc\neg\varphi \leftrightarrow \neg \bigcirc \varphi \tag{21}$$
$$\bigcirc(\varphi \wedge \psi) \leftrightarrow \bigcirc\varphi \wedge \bigcirc\psi \tag{22}$$
$$\bigcirc(\varphi \vee \psi) \leftrightarrow \bigcirc\varphi \vee \bigcirc\psi \tag{23}$$
$$\bigcirc(\varphi \rightarrow \psi) \leftrightarrow (\bigcirc\varphi \rightarrow \bigcirc\psi) \tag{24}$$
$$\varphi \,\mathcal{U}\, \psi \leftrightarrow \psi \vee (\varphi \wedge \bigcirc(\varphi \,\mathcal{U}\, \psi)) \tag{25}$$
$$\varphi \,\mathcal{B}\, \psi \leftrightarrow \psi \wedge (\varphi \vee \bigcirc(\varphi \,\mathcal{B}\, \psi)) \tag{26}$$

Theorems (13)-(15) allow defining $\Box$, $\Diamond$ and $\mathcal{W}$ in terms of $\mathcal{U}$ and $\mathcal{B}$. The formulas (17) and (18) correspond to the De Morgan axioms between operators $\mathcal{U}$ and $\mathcal{B}$. It is easy to see that, together with (13) and (14) they directly imply the corresponding De Morgan axioms (19) and (20) for $\Box$ and $\Diamond$. An important difference with respect to LTL is that, when using these De Morgan axioms, some care must be taken if double negation is involved. For instance, by (19), the formula $\neg\Diamond\neg\varphi$ is equivalent to $\neg\neg\Box\varphi$, but this is not in general equivalent to $\Box\varphi$. A simple counterexample is the interpretation $\langle \overline{H}, \overline{T} \rangle$ with all $T_i = \{p\}$ but some $H_j = \emptyset$, as it satisfies $\neg\neg\Box p$ but not $\Box p$. As a result, we cannot further define $\mathcal{B}$ (resp. $\Box$) in terms of $\mathcal{U}$ (resp. $\Diamond$) or vice versa, as happens in LTL. We have included other LTL standard properties like (21)-(24) to show that the "shifting" behaviour of $\bigcirc$ with respect to classical connectives is the same as in LTL, or (25) and (26) that represent the inductive propagation of $\mathcal{U}$ and $\mathcal{B}$ respectively.

The *(Linear) Temporal Equilibrium Logic* (TEL) is a nonmonotonic version of THT where we establish a models selection criterion. Given two interpretations $M = \langle \overline{H}, \overline{T} \rangle$ and $M' = \langle \overline{H'}, \overline{T'} \rangle$ we say that M is *lower than* M', written $M \leq M'$, when $\overline{T} = \overline{T'}$ and for all $i \geq 0$, $H_i \subseteq H'_i$. As usual, $M < M'$ stands for $M \leq M'$ but $M \neq M'$.

Definition 1 (Temporal Equilibrium Model). *An interpretation M is a* temporal equilibrium model *of a theory Γ if M is a total model of Γ and there is no other model $M' < M$ of Γ.* □

Note that any temporal equilibrium model is total, that is, it has the form $\langle \overline{T}, \overline{T} \rangle$ and so can be actually seen as an interpretation $\overline{T}$ in the standard LTL.

The idea behind temporal equilibrium models is that they capture the answer sets of programs that depend on a fixed temporal index. This is expressed by the following proposition extracted from [19].

Proposition 1. *Let Γ be a combination of non-modal connectives $\wedge, \vee, \neg, \rightarrow, \bot$ with expressions like $\bigcirc^i p$, being p an atom, and let n be the maximum value for i in all $\bigcirc^i p$ occurring in Γ. Then $\langle \overline{T}, \overline{T} \rangle$ is a temporal equilibrium model of Γ iff (1) $T_i = \emptyset$ for all $i > n$; and (2) $\langle X, X \rangle$ with $X = \bigcup_{i=0}^{n} \{\bigcirc^i p \mid p \in T_i\}$ is an equilibrium model of Γ, reading each '$\bigcirc^i p$' as a new atom in the signature.* □

That is, once $\Box, \Diamond, \mathcal{U}$ and $\mathcal{W}$ are removed, we can reduce temporal equilibrium models to (non-temporal) equilibrium models (that is, answer sets) for an extended signature with atoms like $\bigcirc^i p$. To illustrate the effect of this definition, let us consider a pair of examples from [19]. Take, for instance, the theory $\{\Diamond p\}$. This would correspond to an infinite disjunction of the form $p \vee \bigcirc p \vee \bigcirc \bigcirc p \vee \ldots$ or, when making time explicit, to an ASP disjunctive rule like $p(0) \vee p(1) \vee p(2) \vee \ldots$ This generates temporal equilibrium models where p is false in every situation excepting for exactly one index i where p is made true. In other words, the temporal equilibrium models of theory $\{\Diamond p\}$ are captured by the LTL models of the formula $\neg p \, \mathcal{U} \, (p \wedge \bigcirc \Box \neg p)$.

As a second example, take the theory Γ just consisting of the formula $\Box(\neg p \rightarrow \bigcirc p)$. This would informally correspond to an infinite program with the rules $p(1) \leftarrow \neg p(0)$; $p(2) \leftarrow \neg p(1)$; $p(3) \leftarrow \neg p(2)$; etc. It is not difficult to see that the temporal equilibrium models of Γ are captured by the LTL models of $\neg p \wedge \Box(\neg p \leftrightarrow \bigcirc p)$.

In [19] it was shown as an example how TEL can be used to encode the action language $\mathcal{B}$ [14] in a straightforward way.

The definition of strong equivalence for TEL theories is extended from the non-temporal case in an obvious way. An important observation is that proving the equivalence of two temporal theories in THT is a trivial sufficient condition for their strong equivalence under TEL – if they have the same THT models, they will always lead to the same set of temporal equilibrium models, regardless the context. The next section shows how this THT equivalence test can be transformed into provability in standard LTL.

4 Translating THT into LTL

Given a propositional signature Σ, let us denote $\Sigma^* = \Sigma \cup \{p' \mid p \in \Sigma\}$ which is going to be the new propositional signature in LTL. For any temporal formula φ we define its translation φ^* as follows:

1. $\bot^* \stackrel{def}{=} \bot$
2. $p^* \stackrel{def}{=} p'$ for any $p \in \Sigma$
3. $(\odot\varphi)^* \stackrel{def}{=} \odot\varphi^*$, if $\odot \in \{\Box, \Diamond, \bigcirc\}$
4. $(\varphi \odot \psi)^* \stackrel{def}{=} \varphi^* \odot \psi^*$, when $\odot \in \{\wedge, \vee, \mathcal{U}, \mathcal{W}, \mathcal{B}\}$
5. $(\varphi \rightarrow \psi)^* \stackrel{def}{=} (\varphi \rightarrow \psi) \wedge (\varphi^* \rightarrow \psi^*)$

From the last point and the fact that $\neg\varphi = \varphi \rightarrow \bot$, it follows that $(\neg\varphi)^* = (\varphi \rightarrow \bot) \wedge (\varphi^* \rightarrow \bot) = \neg\varphi \wedge \neg\varphi^*$. Similarly, $(\varphi \leftrightarrow \psi)^* = (\varphi \leftrightarrow \psi) \wedge (\varphi^* \leftrightarrow \psi^*)$.

We associate to any THT interpretation $M = \langle \overline{H}, \overline{T} \rangle$ the LTL interpretation $M^t = \overline{I}$ in LTL defined as the sequence of sets of atoms $I_i = \{p' \mid p \in H_i\} \cup T_i$, for any $i \geq 0$. Informally speaking, M^t considers a new primed atom p' per each $p \in \Sigma$ in the original signature. In the LTL interpretation, the primed atom p' represents the fact that p occurs at some point in the $\overline{H}$ component, whereas the original symbol p is used to represent an atom in $\overline{T}$. As a THT interpretation must satisfy $H_i \subseteq T_i$ by construction, we may have LTL interpretations that do not correspond to any THT one. In particular, for an arbitrary $\overline{I}$, we will only be able to form some M such that $M^t = \overline{I}$ when the set of primed atoms at each I_i is a subset of the non-primed ones. In other words, only LTL interpretations $\overline{I}$ satisfying the axiom:

$$\Box(p' \rightarrow p) \tag{27}$$

will have a corresponding THT interpretation M such that $\overline{I} = M^t$.

Example 2. $M = ((\emptyset, \{p, q\}), (\{p\}, \{p, q\}), (\{q\}, \{q\}))$ is a model of the theory $\{\Box(\neg p \rightarrow q) \wedge \Diamond q\}$. In the same way, the corresponding interpretation $M^t = (\{p, q\}, \{p', p, q\}, \{q', q\})$ is a model of

$$\begin{array}{ll} (\Box(\neg p \rightarrow q) \wedge \Diamond q)^* & \leftrightarrow \Box(\neg p \rightarrow q)^* \wedge (\Diamond q)^* \leftrightarrow \\ \Box((\neg p \rightarrow q) \wedge ((\neg p)^* \rightarrow q')) \wedge \Diamond q' & \leftrightarrow \Box((\neg p \rightarrow q) \wedge (\neg p \rightarrow q')) \wedge \Diamond q'. \end{array}$$

In general:

Theorem 1. *Let $M = \langle \overline{H}, \overline{T} \rangle$ be any THT interpretation and φ any formula. For any $i \geq 0$, it holds that*

(a) $\langle \overline{H}, \overline{T} \rangle, i \models \varphi$ *if and only if* $M^t, i \models \varphi^*$ *in LTL; and*
(b) $\langle \overline{T}, \overline{T} \rangle, i \models \varphi$ *if and only if* $M^t, i \models \varphi$ *in LTL.*

Proof. We proceed by induction. For the base case, it trivially holds for $\bot$ whereas for an atom p, we have these equivalence chains

(a) $(\langle \overline{H}, \overline{T} \rangle, i \models p) \Leftrightarrow (p \in H_i) \Leftrightarrow (p' \in I_i) \Leftrightarrow (M^t, i \models p')$.
(b) $(\langle \overline{T}, \overline{T} \rangle, i \models p) \Leftrightarrow (p \in T_i) \Leftrightarrow (p \in I_i) \Leftrightarrow (M^t, i \models p)$

For the inductive step, we detail the proof for the classical connective $\rightarrow$, the (unary) temporal operator $\bigcirc$ and the (binary) temporal operator $\mathcal{U}$. For the classical connectives $\wedge$ and $\vee$ the proof is straightforward; for connective $\mathcal{B}$, it is completely analogous to that for $\mathcal{U}$; and finally, the rest of connectives can be defined in terms of the previous ones.

1. To prove (a) for the implication we have the chain of equivalent conditions:

$$\langle \overline{H}, \overline{T} \rangle, i \models \varphi \rightarrow \psi \overset{def}{\Longleftrightarrow} \left\{ \begin{array}{l} \langle \overline{H}, \overline{T} \rangle, i \not\models \varphi \ or \ \langle \overline{H}, \overline{T} \rangle, i \models \psi \\ and \\ \langle \overline{T}, \overline{T} \rangle, i \not\models \varphi \ or \ \langle \overline{T}, \overline{T} \rangle, i \models \psi \end{array} \right\} \overset{ind}{\Longleftrightarrow}$$
$$\left\{ \begin{array}{l} M^t, i \not\models \varphi^* \ or \ M^t, i \models \psi^* \\ and \\ M^t, i \not\models \varphi \ \ or \ M^t, i \models \psi \end{array} \right\} \Longleftrightarrow \left\{ \begin{array}{l} M^t, i \models (\varphi^* \rightarrow \psi^*) \\ and \\ M^t, i \models (\varphi \rightarrow \psi) \end{array} \right\} \overset{def}{\Longleftrightarrow}$$
$$M^t, i \models (\varphi \rightarrow \psi)^*$$

   For proving (b) it suffices with considering, in each of the three pairs of conjunctive conditions above, only the second conjunct.
2. For operator $\bigcirc$, note that
   (a) $\langle \overline{H}, \overline{T} \rangle, i \models \bigcirc \varphi \overset{def}{\Longleftrightarrow} \langle \overline{H}, \overline{T} \rangle, i+1 \models \varphi \overset{ind}{\Longleftrightarrow} M^t, i+1 \models \varphi^* \overset{def}{\Longleftrightarrow} M^t, i \models \bigcirc \varphi^*$
   (b) $\langle \overline{T}, \overline{T} \rangle, i \models \bigcirc \varphi \overset{def}{\Longleftrightarrow} \langle \overline{T}, \overline{T} \rangle, i+1 \models \varphi \overset{ind}{\Longleftrightarrow} M^t, i+1 \models \varphi \overset{def}{\Longleftrightarrow} M^t, i \models \bigcirc \varphi$
3. Finally, for $\mathcal{U}$ it follows that:
   (a) $\langle \overline{H}, \overline{T} \rangle, i \models \varphi \,\mathcal{U}\, \psi \overset{def}{\Longleftrightarrow}$
   $\exists j \geq i,\ (M, j \models \psi)\ and\ \forall k\ s.t.\ i \leq k < j,\ (M, k \models \varphi) \overset{ind}{\Longleftrightarrow}$
   $\exists j \geq i,\ (M^t, j \models \psi^*)\ and\ \forall k\ s.t.\ i \leq k < j,\ (M, k \models \varphi^*) \overset{def}{\Longleftrightarrow}$
   $M^t, i \models \varphi^* \,\mathcal{U}\, \psi^* \overset{def}{\Longleftrightarrow} M^t, i \models (\varphi \,\mathcal{U}\, \psi)^*$
   (b) $\langle \overline{T}, \overline{T} \rangle, i \models \varphi \,\mathcal{U}\, \psi \overset{def}{\Longleftrightarrow}$
   $\exists j \geq i,\ (M, j \models \psi)\ and\ \forall k\ s.t.\ i \leq k < j,\ (M, k \models \varphi) \overset{ind}{\Longleftrightarrow}$
   $\exists j \geq i,\ (M^t, j \models \psi)\ and\ \forall k\ s.t.\ i \leq k < j\ (M, k \models \varphi) \overset{def}{\Longleftrightarrow}$
   $M^t, i \models \varphi \,\mathcal{U}\, \psi$ □

Theorem 2 (Main theorem). *Let Γ_1 and Γ_2 be a pair of temporal theories, and $\bigwedge \Gamma_1$ and $\bigwedge \Gamma_2$ the conjunctions of their respective sets of formulas. Then Γ_1 and Γ_2 are strongly equivalent with respect to temporal equilibrium models if the formula (27) $\rightarrow (\bigwedge \Gamma_1 \leftrightarrow \bigwedge \Gamma_2)^*$ is valid in LTL.*

Proof. Once the axiom schemata (27) are fixed as hypotheses, Theorem 1 allows us to establish a one to one correspondence between models of $\bigwedge \Gamma_1 \leftrightarrow \bigwedge \Gamma_2$ in THT and models of $(\bigwedge \Gamma_1 \leftrightarrow \bigwedge \Gamma_2)^*$ in LTL. Thus, if $(\bigwedge \Gamma_1 \leftrightarrow \bigwedge \Gamma_2)^*$ is LTL valid, this means that $\bigwedge \Gamma_1$ and $\bigwedge \Gamma_2$ are THT-equivalent, which is a sufficient condition for strong equivalence in TEL. □

As we will show in the next section, we may also use this result to detect a redundant formula φ in some theory Γ. To this aim, we would have to show that Γ and $\Gamma' = \Gamma \setminus \{\varphi\}$ are strongly equivalent. From the theorem above, it follows that:

Corollary 1. *Let Γ be a temporal theory, $\bigwedge \Gamma$ the conjunction of its formulas and φ some arbitrary temporal formula. Then Γ and $\Gamma \cup \{\varphi\}$ are strongly equivalent if the formula (27) $\rightarrow (\bigwedge \Gamma \rightarrow \varphi)^*$ is valid in LTL.* □

5 Back to the Example

When representing Example 1 in TEL, the encoding is quite obvious. As happens in Equilibrium Logic, the logic programming notation is directly replaced by a logical syntax, so that the rule arrow '$\leftarrow$', the comma ',' and the default negation '*not* ' respectively represent standard implication, conjunction and negation. On the other hand, in order to capture the transition rules of program Π_1, only a small subset of the linear temporal syntax is actually required. All the rules depending on the temporal (universally quantified) variable I will be scoped now by a $\Box$ operator, whereas the atoms with argument $I+1$ will be further preceeded by a $\bigcirc$ connective. As a result, we obtain the theory Γ_1 consisting of the formulas:

$$\Box(toggle_j \wedge \overline{sw}_j \rightarrow \bigcirc sw_j) \tag{28}$$
$$\Box(toggle_j \wedge sw_j \rightarrow \bigcirc \overline{sw}_j) \tag{29}$$
$$\Box(toggle_j \wedge sw_j \rightarrow \bigcirc \overline{light}) \tag{30}$$
$$\Box(toggle_1 \wedge \overline{sw}_1 \wedge sw_2 \rightarrow \bigcirc light) \tag{31}$$
$$\Box(toggle_2 \wedge \overline{sw}_2 \wedge sw_1 \rightarrow \bigcirc light) \tag{32}$$
$$\Box(toggle_1 \wedge toggle_2 \rightarrow \bot) \tag{33}$$
$$\Box(f \wedge \neg \bigcirc \overline{f} \rightarrow \bigcirc f) \tag{34}$$
$$\Box(\overline{f} \wedge \neg \bigcirc f \rightarrow \bigcirc \overline{f}) \tag{35}$$
$$\Box(f \wedge \overline{f} \rightarrow \bot) \tag{36}$$

that respectively correspond to the rules (11)-(9).

In order to answer questions Q1 and Q2 proposed in Section 2, we have implemented the φ^* translation[4] and fed the `LWB` linear temporal logic module with the resulting formulas extracted from Corollary 1 to detect redundant rules. In particular, regarding question Q1, we have been able to prove that, given the theory Γ_2 consisting of Γ_1 plus:

$$\Box(sw_1 \wedge sw_2 \rightarrow light) \tag{37}$$
$$\Box(\overline{sw}_1 \rightarrow \overline{light}) \tag{38}$$
$$\Box(\overline{sw}_2 \rightarrow \overline{light}) \tag{39}$$

(that captures *light* as an indirect effect) then rule (30), that specified when *light* became off depending on each $toggle_j$, becomes redundant and can be safely removed from Γ_2. In fact, we could just prove that (30) followed from the formulas (29), (38) and (39).

However, when we consider the effect axioms (31),(32) we used to specify when the *light* becomes on, our checker just provides a negative answer. Since we are just dealing with a sufficient condition for strong equivalence, this does not provide any concluding information about Q1 for these rules. Fortunately, in this case it is not difficult to see that, in fact, these rules cannot be removed

[4] Available at `http://www.dc.fi.udc.es/~cabalar/eqwb.html`

in any context without changing the general behaviour of the program. To see why, just think about adding, as a new effect of $toggle_1$, that switch sw_2 is disconnected:

$$\Box(toggle_1 \wedge sw_2 \rightarrow \overline{sw}_2)$$

The addition of this new rule would clearly make a different effect depending on (31) is present in the theory or not. In particular, when we perform $toggle_1$ in a situation in which $\overline{sw_1} \wedge sw_2$, if we do not have (31) the light will just remain off, whereas if this formula is included, we will get no equilibrium model.

In order to answer question Q2, we considered theory $\Gamma_3 = \Gamma_2 \setminus \{(3) - (5)\}$ and tried to see whether (34),(35) were redundant, for the case $f = light$. Although it could seem that $light$ value is "defined" in terms of sw_1 and sw_2 by rules (37)-(39), we actually obtained a negative answer, so no conclusion is obtained. It is easy to see that, in this case, we can find situations in which removing inertia may cause different effects depending on the context. For instance, if we had $light$ and do not provide information for sw_1 and sw_2 or their explicit negations in a given state, the inertia would maintain $light$ in the next situation (toggling yields no effect). If we remove inertia, however, neither $light$ nor $\overline{light}$ would hold in the next state.

If we look for a stronger relation between $light$ and sw_1, sw_2 we can move to consider Γ_3, consisting of Γ_2 and the rules:

$$\Box(light \rightarrow sw_1) \qquad \Box(light \rightarrow sw_2) \qquad \Box(\neg light \rightarrow \overline{light})$$

we finally obtain that inertia (34),(35) for $f = light$ is redundant. The explanation for this is simple – note that these rules, together with (37) allow concluding (among other things) that $\Box(light \leftrightarrow sw_1 \wedge sw_2)$ which is a double implication, actually defining $light$ as the conjunction of sw_1 and sw_2 in any situation.

6 Conclusions

In this paper we have studied the problem of strong equivalence of logic programs that represent transition systems. To this aim, we have incorporated the set of modal operators typically used in Linear Temporal Logic (LTL) into logic programs, and applied a previously introduced [19] general semantics for any arbitrary (propositional) temporal theory. This semantics is a temporal extension of Equilibrium Logic [7] (a logical characterisation of stable models) and its monotonic basis, the logic of Here-and-There (HT). As a result, we obtained a sufficient condition for checking strong equivalence of temporal logic programs and we implemented a method for reducing this condition to provability in standard LTL. The advantage of this process is that, to the best of our knowledge, it constitutes the first automated tool for guaranteeing strong equivalence of logic programs representing transition systems, as the previous existing checkers either require fixing a numerical path length to obtain a ground program, or in the case of programs with variables [15], cannot properly deal with transition

systems. There exists, however, a very recent prover (QHT [16]) for strong equivalence of programs with variables, which relies on a method proposed in [17] for reducing Quantified Here and There to first order classical logic, allowing in this way, the use of a classical theorem prover as a backend. Unlike [15], QHT allows dealing with functions and so, could potentially handle situation indices with a Peano-like successor function. We leave for future work checking this alternative, although in a general case, the currently presented reduction to LTL has the obvious advantage of handling a decidable inference method.

Of course, the main current drawback of our strong equivalence test is that, when it provides a negative answer, no conclusion can be drawn, since we have not proved yet whether it also constitutes a necessary condition for strong equivalence, something that was obtained [8] for the non-modal case with HT-equivalence. The desirable situation would be, instead, that a negative answer came with a counterexample, that is, a piece of (temporal) program that added to the original ones to be tested yields different effects in each case. This point is left for the immediate future work.

Acknowledgements. We wish to thank David Pearce, Agustín Valverde, Manuel Ojeda, Alfredo Burrieza for their helpful discussions about this work. We are also thankful to the anonymous referees for their comments and observations.

References

1. Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: The Logic Programming Paradigm: a 25-Year Perspective, pp. 169–181. Springer, Heidelberg (1999)
2. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence 25, 241–273 (1999)
3. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R.A., Bowen, K.A. (eds.) Logic Programming: Proc.of the Fifth International Conference and Symposium, vol. 2, pp. 1070–1080. MIT Press, Cambridge (1988)
4. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving, pp. 285–316. Elsevier, Amsterdam (2003)
5. Gelfond, M.: Answer Sets. In: Handbook of Knowledge Representation, pp. 285–316. Elsevier, Amsterdam (2007)
6. WASP: ASP solvers web page (last update, 2005), http://dit.unitn.it/~wasp/Solvers/index.html
7. Pearce, D.: A new logical characterisation of stable models and answer sets. In: Dix, J., Przymusinski, T.C., Moniz Pereira, L. (eds.) NMELP 1996. LNCS, vol. 1216. Springer, Heidelberg (1997)
8. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. Computational Logic 2(4), 526–541 (2001)
9. Pearce, D., Valverde, A.: Towards a first order equilibrium logic for nonmonotonic reasoning. In: Alferes, J.J., Leite, J.A. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 147–160. Springer, Heidelberg (2004)

10. Ferraris, P., Lee, J., Lifschitz, V.: A new perspective on stable models. In: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 372–379 (2004)
11. McCarthy, J., Hayes, P.: Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence Journal 4, 463–512 (1969)
12. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. Artificial Intelligence Journal 153, 49–104 (2004)
13. Ferraris, P.: Causal theories as logic programs. In: Proc. of the 20th Workshop on Logic Programming (WLP 2006) (2006)
14. Gelfond, M., Lifschitz, V.: Action languages. Linköping Electronic Articles in Computer and Information Science 3(16) (1998)
15. Eiter, T., Faber, W., Traxler, P.: Testing strong equivalence of datalog programs - implementation and examples. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 437–441. Springer, Heidelberg (2005)
16. Cabalar, P.: QHT - a prover for quantified here and there (2008), http://www.dc.fi.udc.es/~cabalar/eqwb.html
17. Lifschitz, V., Pearce, D., Valverde, A.: A characterization of strong equivalence for logic programs with variables. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483. Springer, Heidelberg (2007)
18. Heyting, A.: Die formalen Regeln der intuitionistischen Logik. Sitzungsberichte der Preussischen Akademie der Wissenschaften. Physikalisch-mathematische Klasse, 42–56 (1930)
19. Cabalar, P., Vega, G.P.: Temporal equilibrium logic: a first approach. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) EUROCAST 2007. LNCS, vol. 4739, pp. 241–248. Springer, Heidelberg (2007)
20. Kamp, J.A.: Tense Logic and the Theory of Linear Order. PhD thesis, University of California at Los Angeles (1968)
21. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer, Heidelberg (1991)
22. Pearce, D., Tompits, H., Woltran, S.: Encodings of equilibrium logic and logic programs with nested expressions. In: Bradzil, P., Jorge, A. (eds.) EPIA 2001. LNCS (LNAI), vol. 2258, pp. 306–320. Springer, Heidelberg (2001)
23. Lin, F.: Reducing strong equivalence of logic programs to entailment in classical propositional logic. In: Bradzil, P., Jorge, A. (eds.) Proc. of the 8th Intl. Conf. on Principles and Knowledge Representation and Reasoning (KR 2002), pp. 170–176. Morgan Kaufmann, San Francisco (2002)
24. Heuerding, A., Jäger, G., Schwendimann, S., Seyfried, M.: A logics workbench. AI Communications 9(2), 53–58 (1996)
25. Thielscher, M.: Ramification and causality. Artificial Intelligence Journal 89(1–2), 317–364 (1997)
26. Valverde, A.: tabeql: A tableau based suite for equilibrium logic. In: Alferes, J.J., Leite, J.A. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 734–737. Springer, Heidelberg (2004)
27. Chen, Y., Lin, F., Li, L.: SELP - a system for studying strong equivalence between logic programs. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 442–446. Springer, Heidelberg (2005)

Consistency Preservation and Crazy Formulas in BMS*

Guillaume Aucher

University of Otago (NZ) - Université Paul Sabatier (F)
IRIT, 118 route de Narbonne, 31062 Toulouse cedex 9 (F)
aucher@irit.fr

Abstract. We provide conditions under which seriality is preserved during an update in the BMS framework. We consider not only whether the entire updated model is serial but also whether its generated submodels are serial. We also introduce the notion of crazy formulas which are formulas such that after being publicly announced at least one of the agents' beliefs become inconsistent.

1 Introduction

Providing formalisms which allow agents to reason adequately about belief and belief change is an important goal in artificial intelligence so that artificial agents can act autonomously and rationally in a given environment. An obvious requirement for these agents should be that their beliefs always remain consistent whatever happens.

One way to represent formally the agents' beliefs about a given (static) situation is by means of an epistemic model. Expressing that the agents' beliefs are consistent amounts to assume that the accessibility relations of the epistemic model are serial.

The next step is to introduce events and to model their effects on the agents' beliefs. An influential formalism has been proposed in dynamic epistemic logic by Baltag, Moss and Solecki (to which we will refer by the term BMS [3]) that deals with this issue. Their idea is to represent how an event occurring in this situation is perceived by the agents by means of an event model and then to define a formal update mechanism that specifies how the agents update their beliefs according to this event model and the original epistemic model. This yields a new epistemic model corresponding to the resulting situation. However, as it turns out, it is quite possible formally that this new epistemic model is not serial, even if the original epistemic model and the event model were serial. For example, if agent A believes $\neg\phi$ then after a public announcement of ϕ agent A's accessibility relation is not serial anymore. Specifying formally under which conditions this happens in general (and not only for public announcements) has not been studied so far. In this paper we tackle this issue and also introduce some

* I thank my supervisors Hans van Ditmarsch and Andreas Herzig for useful comments on this paper.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 21–33, 2008.

formulas, called 'crazy formulas', such that some of the agents' beliefs always become inconsistent after they are publicly announced: they become 'crazy'.

The paper is organized as follows. In Section 2, we recall the BMS system together with the definitions of seriality and generated (sub)model. In Section 3, we provide conditions under which the entire updated epistemic model is serial and introduce what we call 'crazy formulas'. In Section 4, we investigate under which conditions a generated submodel of the entire updated epistemic model is serial. Finally, we conclude in Section 5.

2 The BMS System

In this paper, Φ is a set of propositional letters and G is a finite set of agents.

2.1 Epistemic Model and Generated Submodel

Epistemic model. An epistemic model is just a particular kind of Kripke model [5] where instead of having a single accessibility relation we have a set of accessibility relations, one for each agent.

Definition 1. *An* epistemic model M *is a triple* $M = (W, R, V)$ *such that*

- W *is a non-empty set of possible worlds;*
- $R : G \to 2^{W \times W}$ *assigns an accessibility relation to each agent;*
- $V : \Phi \to 2^W$ *assigns a set of possible worlds to each propositional letter and is called a valuation.*

If $M = (W, R, V)$ *is an epistemic model, a pair* (M, w_a) *with* $w_a \in W$ *is called a* pointed epistemic model. *We also write* $R_j = R(j)$ *and* $R_j(w) = \{w' \in W \mid wR_jw'\}$, *and* $w \in M$ *for* $w \in W$.

Intuitively, a pointed epistemic model (M, w_a) represents from an external point of view how the actual world w_a is perceived by the agents G. The possible worlds W are the relevant worlds needed to define such a representation and the valuation V specifies which propositional facts (such as 'it is raining') are true in these worlds. Finally the accessibility relations R_j model the notion of belief. We set $w' \in R_j(w)$ in case the world w' is compatible with agent j's belief in world w. We can then define the notion of seriality for epistemic models.

Definition 2. *Let* $M = (W, R, V)$ *be an epistemic model. We say that* M *is* serial *when for all* $j \in G$, R_j *satisfies the following condition:*

Seriality: for all $w \in W$, $R_j(w) \neq \emptyset$.

Intuitively, an epistemic model which is not serial means that in the corresponding situation there is an agent j whose beliefs are inconsistent. More precisely, it means that it is not common belief that the agents' beliefs are consistent.

Now inspiring ourselves from modal logic, we can define a language for epistemic models. The modal operator is just replaced by a 'belief' operator, one for each agent.

Definition 3. *The language $\mathcal{L}^U$ is defined as follows:*

$$\mathcal{L}^U : \phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid B_j\phi \mid U\phi$$

where p ranges over Φ and j over G. Moreover, $\phi \vee \phi'$ is an abbreviation for $\neg(\neg\phi \wedge \neg\phi')$; $\phi \rightarrow \phi'$ is an abbreviation for $\neg\phi \vee \phi'$; $\hat{B}_j\phi$ is an abbreviation for $\neg B_j\neg\phi$; $O\phi$ is an abbreviation for $\neg U\neg\phi$; and $\bot$ is an abbreviation for $\neg\top$.

Finally, by $\mathcal{L}$ we denote the language $\mathcal{L}^U$ without the universal modality U.

Intuitively, $B_j\phi$ means that agent j believes that the formula ϕ is true. U is the universal modality which is introduced here only for technical reasons in order to express the seriality preservation conditions. Now we can give a genuine meaning to the formulas of this language by defining truth conditions for these formulas on the class of epistemic models.

Definition 4. *Let $M = (W, R, V)$ be an epistemic model and $w \in W$. $M, w \models \phi$ is defined inductively as follows:*

$$\begin{array}{lll}
M, w \models \top & & \\
M, w \models p & \textit{iff} & w \in V(p) \\
M, w \models \neg\phi & \textit{iff} & \textit{not } M, w \models \phi \\
M, w \models \phi \wedge \phi' & \textit{iff} & M, w \models \phi \textit{ and } M, w \models \phi' \\
M, w \models B_j\phi & \textit{iff} & \textit{for all } v \in R_j(w), M, v \models \phi \\
M, w \models U\phi & \textit{iff} & \textit{for all } v \in W, M, v \models \phi
\end{array}$$

We write $M \models \phi$ for $M, w \models \phi$ for all $w \in M$. If C is a class of epistemic models, we write $\models_{\mathsf{C}} \phi$ when for all $M \in \mathsf{C}$, $M \models \phi$. Finally, we write $\models \phi$ when for all epistemic model M, $M \models \phi$.

So agent j believes ϕ in world w (formally $M, w \models B_j\phi$) if ϕ is true in all the worlds that agent j considers possible (in world w). $M, w \models U\phi$ expresses that ϕ is valid in the model M. The universal modality is thus a stronger notion than the common belief modality often used in epistemic logic.

Example 1. We take up the coin example of [3]. Assume there are two agents Ann and Bob who are in a room where there is a coin in a box. The coin is actually heads up but the box is closed. So both of them do not know whether the coin is heads or tails up. Now assume that Bob cheats and looks at the coin, Ann suspecting nothing about it. This resulting situation is modeled in the pointed epistemic model (M, w_a) of Figure 1. The accessibility relations are represented by arrows indexed by A (standing for *A*nn) or B (standing for *B*ob); p stands for 'the coin is heads up' and the boxed world w_a stands for the actual world. Now thanks to the language $\mathcal{L}$ we can express what is true in this situation. For example, Bob correctly believes that the coin is heads up: $M, w_a \models p \wedge B_B p$; while Ann believes that he does not know whether the coin is heads or tails up: $M, w_a \models B_A(\neg B_B p \wedge \neg B_B \neg p)$.

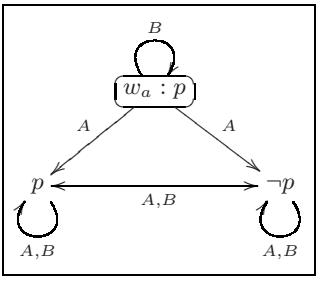


Fig. 1. The coin example

Generated submodel. An epistemic model might contain some information that is not relevant to model a given situation. We now define the notion of generated submodel that discards this useless information.

Definition 5. *Let* $M = (W, R, V)$ *and* $M' = (W', R', V')$ *be two epistemic models and* $w_a \in W$.

- *We say that* M' *is a* submodel *of* M *if* $W' \subseteq W$*; for all* $j \in G$, $R'_j = R_j \cap (W' \times W')$ *and for all* $p \in \Phi$, $V'(p) = V(p) \cap W'$. *We also say that* M' *is the restriction of* M *to* W'.
- *The* submodel of M generated by w_a *is the restriction of* M *to* $(\bigcup_{j \in G} R_j)^*(w_a)$[1]. *In case the submodel of* M *generated by* w_a *is* M *itself, we say that* M is generated by w_a *and that* w_a *is the root of* M.

Proposition 1. *Let* $M = (W, R, V)$ *be an epistemic model and* M' *a submodel of* M *generated by some* $w_a \in W$. *Then for all* $w \in M'$ *and all* $\phi \in \mathcal{L}$, $M, w \models \phi$ *iff* $M', w \models \phi$.

This proposition entails that in a pointed epistemic model (M, w_a) where w_a stands for the actual world, the part of the model M that is really relevant for us to model the corresponding situation is the submodel of M generated by w_a.

2.2 Event Model

Epistemic models are used to model how the agents perceive the actual world in terms of beliefs about the world and about the other agents' beliefs. The insight of the BMS approach is that one can describe how an event is perceived by the agents in a very similar way. Indeed, the agents' perception of an event can also be described in terms of beliefs: for example, while Bob looks at the coin and sees that it is heads up (event a_a) Ann *believes* that nothing happens (event b). This leads them to define the notion of event model whose definition is very similar to that of an epistemic model.

[1] If R is a relation, R^+ is defined by $R^+(w) = \{v |$ there is $w_1, \ldots, w_n = v$ such that $w_i R w_{i+1}\}$. R^* is defined by $R^*(w) = \{w\} \cup R^+(w)$. See [5].

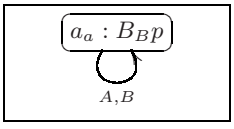


Fig. 2. Public announcement of $B_B p$

Definition 6. *An* event model A *is a triple* $A = (E, R, Pre)$ *such that*

- E *is a finite and non-empty set of possible events;*
- $R : G \to 2^{E \times E}$ *assigns an accessibility relation to each agent;*
- $Pre : E \to \mathcal{L}$ *assigns an epistemic formula to each possible event.*

If $A = (E, R, Pre)$ *is an event model, a pair* (A, a_a) *where* $a_a \in E$ *is called a* pointed event model. *We also write* $R_j = R(j)$ *and* $R_j(a) = \{b \in E \mid aR_jb\}$, *and* $a \in A$ *for* $a \in E$.

The main difference with the definition of an epistemic model is that we no longer have a valuation V but instead a function Pre. This function is supposed to specify under which condition an event can physically take place in a possible world.

Example 2. Assume that an external agent announces publicly that Bob believes that the coin is heads up (formally $B_B p$). This event is depicted in Figure 2. There, a_a stands for 'the external agent truthfully announces that Bob believes that the coin is heads up'. Because this event is correctly perceived by Ann and Bob, a_a is the only event considered possible by them. Finally, for this truthful announcement to be made in a possible world, Bob has indeed to believe that the coin is heads up in this world ($B_B p$).

2.3 Product Update

Now, in reality after (or during) this event e, the agents update their beliefs by taking into account these two pieces of information: the event e and the initial situation s. This gives rise to a new situation $s \times e$. This actual update is rendered formally by the following mathematical update product between a pointed epistemic model and a pointed event model.

Definition 7. *Let* $M = (W, R, V, w_a)$ *be a pointed epistemic model and* $A = (E, R, Pre, a_a)$ *a pointed event model such that* $M, w_a \models Pre(a_a)$. *We define their* update product *to be the pointed epistemic model* $M \otimes A = (W \otimes E, R', V', w'_a)$ *where*

1. $W \otimes E = \{(w, a) \in W \times E \mid M, w \models Pre(a)\}$;
2. $(v, b) \in R'_j(w, a)$ *iff* $v \in R_j(w)$ *and* $b \in R_j(a)$;
3. $V'(p) = \{(w, a) \in W \otimes E \mid w \in V(p)\}$;
4. $w'_a = (w_a, a_a)$.

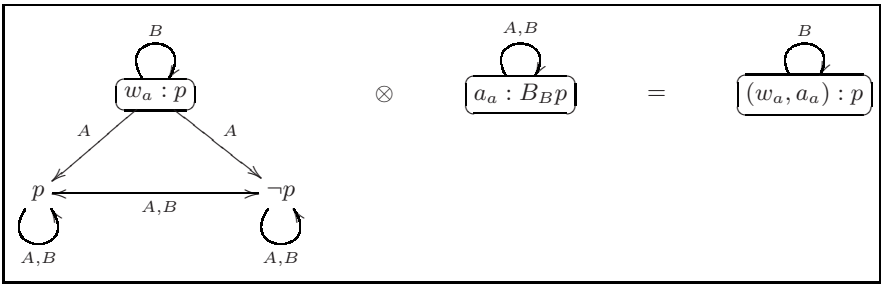


Fig. 3. Failure of seriality preservation

Example 3. This example shows that seriality might not be preserved during an update. If we update the epistemic model depicted in Figure 1 by the truthful public announcement that Bob believes that the coin is heads up (formally $B_B p$) depicted in Figure 2 then we get the epistemic model depicted on the right of Figure 3 where Ann's accessibility relation is not serial.

3 Seriality Preservation for the Entire BMS Product

3.1 Theory

First of all, for a given epistemic model M and a given event model A, we say that the update product $M \otimes A$ is *defined* if there is $w \in M$ and $a \in A$ such that $M, w \models Pre(a)$. We introduce this definition because seriality of updated models makes sense only for defined updated models.

Proposition 2. *Let A be a serial event model and let M be an epistemic model.*

$$M \otimes A \text{ is defined and serial iff}$$
$$M \models O\left(\bigvee_{a\in A} Pre(a)\right) \wedge U \bigwedge_{a\in A}\left(Pre(a) \rightarrow \bigwedge_{j\in G} \hat{B}_j \bigvee_{b\in R_j(a)} Pre(b)\right).$$

Proof. $M \models O\left(\bigvee_{a\in A} Pre(a)\right)$ clearly means that the model $M \otimes A$ is defined. Now it remains to prove that $M \otimes A$ is serial iff

$$M \models U \bigwedge_{a\in A}\left(Pre(a) \rightarrow \bigwedge_{j\in G} \hat{B}_j \bigvee_{b\in R_j(a)} Pre(b)\right).$$

– Assume that $M \models U \bigwedge_{a\in A}\left(Pre(a) \rightarrow \bigwedge_{j\in G} \hat{B}_j \bigvee_{b\in R_j(a)} Pre(b)\right)$ (*). Let $(w, a) \in M \otimes A$ and $j \in G$. Then $M, w \models Pre(a)$. So $M, w \models \bigwedge_{j\in G} \hat{B}_j \bigvee_{b\in R_j(a)} Pre(b)$ by (*). Then $M, w \models \hat{B}_j \bigvee_{b\in R_j(a)} Pre(b)$. So there is $v \in R_j(w)$ and $b \in R_j(a)$ such that $M, v \models Pre(b)$. Then there is $(v, b) \in M \otimes A$ such that $(v, b) \in R_j(w, a)$ by definition of $M \otimes A$. So $M \otimes A$ is serial.

– Assume that $M \not\models U \bigwedge_{a\in A}\left(Pre(a) \rightarrow \bigwedge_{j\in G}\hat{B}_j \bigvee_{b\in R_j(a)} Pre(b)\right)$. Then there is $w \in M$ and $a \in A$ such that $M, w \models Pre(a) \wedge \left(\bigvee_{j\in G} B_j \bigwedge_{b\in R_j(a)} \neg Pre(b)\right)$. Then there is $j \in G$ such that $M, w \models B_j \bigwedge_{b\in R_j(a)} \neg Pre(b)$ (**). So $(w, a) \in M \otimes A$ but there is no $v \in R_j(w)$ and $b \in R_j(a)$ such that $(v, b) \in R_j(w, a)$. Indeed, otherwise we would have $M, w \models \hat{B}_j \bigvee_{b\in R_j(a)} Pre(b)$, which contradicts (**). So $M \otimes A$ is not serial.

We write $\mathcal{S}(A) = O\left(\bigvee_{a\in A} Pre(a)\right) \wedge U \bigwedge_{a\in A}\left(Pre(a) \rightarrow \bigwedge_{j\in G}\hat{B}_j \bigvee_{b\in R_j(a)} Pre(b)\right)$.

$O\left(\bigvee_{a\in A} Pre(a)\right)$ expresses that the updated model $M \otimes A$ is defined.

$U \bigwedge_{a\in A}\left(Pre(a) \rightarrow \bigwedge_{j\in G}\hat{B}_j \bigvee_{b\in R_j(a)} Pre(b)\right)$ expresses that the updated model $M \otimes A$ is serial. Note that the seriality conditions bear only on M and that M does not need to be serial for the proposition to hold. But of course if M is serial then the proposition still holds. However, if the event model is not serial then one cannot get a serial updated model. From this proposition we can easily prove the following corollary.

Corollary 1. *Let C be a class of epistemic models and A a serial event model.*

$$\models_{\mathsf{C}} \neg\mathcal{S}(A)$$

iff there is no epistemic model $M \in \mathsf{C}$ such that $M \otimes A$ is defined and serial.

In other words this corollary tells us under which condition, for a given event model A, whatever epistemic model M we chose, $M \otimes A$ will not be defined or not serial. If this condition is fulfilled that would mean intuitively that in any epistemic situation, if the event (corresponding to this event model) is performed, then afterwards in any case (some of) the agents' beliefs are inconsistent. This is of course counter intuitive and we should then avoid such kinds of event (models).

3.2 Crazy Formulas

We are going to give an example of a class of epistemic formulas such that after they are publicly announced some of the agents' beliefs become inconsistent.

Definition 8. *A* crazy formula *is a satisfiable formula $\phi \in \mathcal{L}$ such that*

$$\models \phi \rightarrow \bigvee_{j\in G} B_j \neg\phi.$$

Proposition 3. *Let ϕ be a crazy formula and let A be the event model corresponding to the public announcement of ϕ. Then there is no epistemic model M such that $M \otimes A$ is defined and serial.*

Proof. Thanks to Corollary 1 it suffices to prove that $\models \neg\mathcal{S}(A)$ i.e. $\models O\phi \rightarrow O\left(\phi \wedge \bigvee_{j\in G} B_j\neg\phi\right)$ because $\mathcal{S}(A) = O\phi \wedge U\left(\phi \rightarrow \bigwedge_{j\in G} \hat{B}_j\phi\right)$. Let M be an epistemic model such that $M \models O\phi$. Let $w \in M$ such that $M, w \models \phi$. Then by definition of a crazy formula $M, w \models \phi \wedge \bigvee_{j\in G} B_j\neg\phi$. Then $M, w \models \phi \wedge \bigvee_{j\in G} B_j\neg\phi$ i.e. $M \models O\left(\phi \wedge \bigvee_{j\in G} B_j\neg\phi\right)$.

Proposition 4. *$\phi = \psi \wedge B_i\neg\psi$, where $\psi \in \mathcal{L}$ is a satisfiable formula, is a crazy formula.*

Proof. One can easily show that $\models \phi \rightarrow B_i\neg\phi$.

We can compare this notion of crazy formula with the notion of selfrefuting and successful formulas studied in [7]. Selfrefuting formulas are formulas that are no longer true after they are publicly announced. An example of such formulas is Moore's sentence $p\wedge\neg B_j p$: if it is announced then p becomes common belief and in particular $B_j p$ becomes true. Here our formulas are a bit different: after they are publicly announced some of the agents' beliefs become inconsistent. On the other hand, successful formulas are formulas which are always true after being publicly announced. One can show that crazy formulas are not successful.

4 Seriality Preservation for Generated Submodels

One should note that it is quite possible that an updated model consists of several disjoint epistemic models. But in practice, as we said in Section 2.1, the epistemic model we are really interested in is the submodel of the entire updated model generated by the actual world (w_a, a_a) . So, more generally, we would like to know under which conditions a particular generated submodel of the entire updated model is serial. That is what we are going to investigate now.

Definition 9. *Let A be an event model, $a \in A$ and $n \in \mathbb{N}$. We define $\delta^n(a)$ inductively as follows.*

- $\delta^0(a) = Pre(a)$;
- $\delta^{n+1}(a) = \delta^0(a) \wedge \bigwedge_{j\in G} \hat{B}_j \bigvee_{b\in R_j(a)} \delta^n(b) \wedge \bigwedge_{j\in G} B_j \bigwedge_{b\in R_j(a)} (Pre(b) \rightarrow \delta^n(b))$.

Intuitively, $M, w \models \delta^n(a)$ means that the submodel of $M\otimes A$ generated by (w, a) is defined and serial up to modal depth n. This interpretation is endorsed by the following two lemmas which will be used to prove the main proposition.

Lemma 1. *Let M be an epistemic model and let A be an event model. For all $w \in M$, $a \in A$, $n \in \mathbb{N}$,*

$M, w \models \delta^{n+1}(a)$ *iff*

$M, w \models \delta^1(a)$ and for all $v \in M$ such that $w = w_0 R_{j_1} w_1 R_{j_2} \dots R_{j_n} w_n = v$ such that there are $a = a_0 R_{j_1} a_1 R_{j_2} \dots R_{j_n} a_n = b$ such that for all $i \in \{0, \dots, n\}$, $M, w_i \models Pre(a_i)$,

$$M, v \models \bigwedge_{j \in G} \hat{B}_j \bigvee_{c \in R_j(b)} Pre(c)$$

Proof. We prove it by induction on n. The case $n = 0$ is clear. We prove the induction step. Assume the property is true for n.

- Assume $M, w \models \delta^{n+2}(a)$. Then $M, w \models \delta^1(a)$ because $\delta^{n+1}(b) \to Pre(b)$ and $\delta^1(a) = Pre(a) \wedge \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} Pre(b)$. Let $v \in M$ such that $w = w_0 R_{j_1} w_1 R_{j_2} \dots R_{j_{n+1}} w_{n+1} = v$ and such that there are $a = a_0 R_{j_1} a_1 R_{j_2} \dots R_{j_{n+1}} a_{n+1} = b$ such that for all $i \in \{0, \dots, n+1\}$, $M, w_i \models Pre(a_i)$.

  By assumption, $M, w \models \bigwedge_{j \in G} B_j \bigwedge_{b \in R_j(a)} (Pre(b) \to \delta^{n+1}(b))$. So $M, w_1 \models \bigwedge_{b \in R_{j_1}(a)} (Pre(b) \to \delta^{n+1}(b))$. Besides $a_1 \in R_{j_1}(a)$ and $M, w_1 \models Pre(a_1)$. So $M, w_1 \models \delta^{n+1}(a_1)$.

  Then, by induction hypothesis, for all v' such that $w_1 = w'_1 R_{j_2} \dots R_{j_{n+1}} w'_{n+1} = v'$ such that there are $a_1 = a'_1 R_{j_2} \dots R_{j_{n+1}} a'_{n+1} = a'$ such that for all i, $M, w'_i \models Pre(a'_i)$,

  $$M, v' \models \bigwedge_{j \in G} \hat{B}_j \bigvee_{b' \in R_j(a')} Pre(b').$$

  So $M, v \models \bigwedge_{j \in G} \hat{B}_j \bigvee_{c \in R_j(b)} Pre(c)$
- Assume $M, w \models \delta^1(a)$ and assume that for all $v \in M$ such that $w = w_0 R_{j_1} \dots R_{j_n} w_{n+1} = v$ such that there are $a = a_0 R_{j_1} \dots R_{j_n} a_{n+1} = b$ such that for all i, $M, w_i \models Pre(a_i)$,

  $$M, v \models \bigwedge_{j \in G} \hat{B}_j \bigvee_{c \in R_j(b)} Pre(c).$$

  Now, assume $M, w \not\models \delta^{n+2}(a)$. Then $M, w \models \neg Pre(a) \vee \left(\bigvee_{j \in G} B_j \bigwedge_{b \in R_j(a)} \neg \delta^{n+1}(b) \right) \vee \left(\bigvee_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} (Pre(b) \wedge \neg \delta^{n+1}(b)) \right)$.
  - $M, w \models \neg Pre(a)$ is impossible by assumption.
  - Assume $M, w \models \bigvee_{j \in G} B_j \bigwedge_{b \in R_j(a)} \neg \delta^{n+1}(b)$. Then for some $i \in G$, $M, w \models B_i \bigwedge_{b \in R_i(a)} \neg \delta^{n+1}(b)$. Then for all $v \in R_i(w)$ and all $b \in R_i(a)$, $M, v \models \neg \delta^{n+1}(b)$ (*).

But by assumption $M, w \models \delta^1(a)$, i.e. $M, w \models Pre(a) \wedge \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} Pre(b)$. Then $M, w \models \hat{B}_i \bigvee_{b \in R_i(a)} Pre(b)$, i.e. there is $v \in R_i(w)$ and $b \in R_i(a)$ such that $M, v \models Pre(b)$ (1).

So $M, v \models \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} Pre(b)$ (2) by assumption (take $w_1 = \ldots = w_n = v$ and $a_1 = \ldots = a_n = b$).

Then by (1) and (2) we get $M, v \models \delta^1(b)$.

Besides, by assumption and because wR_iv and aR_ib, for all u such that $v = v_0 R_{j_1} \ldots R_{j_n} u$ such that there are $b = b_0 R_{j_1} \ldots R_{j_n} b_n = c$ such that for all i $M, v_i \models Pre(b_i)$

$$M, u \models \bigwedge_{j \in G} \hat{B}_j \bigvee_{d \in R_j(c)} Pre(d).$$

So $M, v \models \delta^{n+1}(b)$ by induction hypothesis. This is impossible by (*).

- Assume $M, w \models \bigvee_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} (Pre(b) \wedge \neg\delta^{n+1}(b))$.

Then there is $i \in G$, $v \in R_i(w)$ and $b \in R_i(a)$ such that $M, v \models Pre(b) \wedge \neg\delta^{n+1}(b)$.

By the same argument as above we get to a contradiction.

So finally $M, w \models \delta^{n+2}(a)$.

Lemma 2. *Let M be a finite epistemic model and A be a finite serial event model. Let $n = |M| \cdot |A|$.*[2] *For all $w \in M$ and $a \in A$ such that $M, w \models Pre(a)$,*

1. *$R_j(w, a) \neq \emptyset$ for all $j \in G$ iff $M, w \models \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} Pre(b)$;*
2. *$(v, b) \in \left(\bigcup_{j \in G} R_j \right)^+ (w, a)$ iff there are $w = w_0 R_{j_1} w_1 R_{j_2} \ldots R_{j_n} w_{n-1} = v$ and $a = a_0 R_{j_1} a_1 R_{j_2} \ldots R_{j_n} a_{n-1} = b$ such that for all i, $M, w_i \models Pre(a_i)$.*

Proof. 1. Assume $M, w \models \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} Pre(b)$. Then for all $j \in G$, there is $v \in R_j(w)$ and $b \in R_j(a)$ such that $M, v \models Pre(b)$. Then, by definition of the product update, for all j, there is $(v, b) \in M \otimes A$ such that $(v, b) \in R_j(w, a)$. So for all $j \in G$, $R_j(w, a) \neq \emptyset$.

Assume $M, w \not\models \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} Pre(b)$. Then there is $j \in G$ such that for all $v \in R_j(w)$ and for all $b \in R_j(a)$, $M, v \not\models Pre(b)$. Then, by definition of the product update, there is no $(v, b) \in M \otimes A$ such that $(v, b) \in R_j(w, a)$. So $R_j(w, a) = \emptyset$ for some $j \in G$.

2. $M \otimes A$ is of cardinality at most n due to our hypothesis that $n = |M| \cdot |A|$. So every world $(v, b) \in M \otimes A$ such that $(v, b) \in \left(\bigcup_{j \in G} R_j \right)^+ (w, a)$ is accessible from (w, a) in at most $n - 1$ steps. So,

[2] $|M|$ (resp. $|A|$) is the number of possible worlds (resp. events) of M (resp. A).

$(v,b) \in \left(\bigcup_{j\in G} R_j\right)^+ (w,a)$ iff
there are $j_1,\dots,j_{n-1}$ and $(w_1,a_1),\dots,(w_{n-1},a_{n-1}) \in M \otimes A$ such that $(w,a)R_{j_1}(w_1,a_1)R_{j_2}\dots R_{j_{n-1}}(w_{n-1},a_{n-1}) = (v,b)$ iff
there are $w = w_0R_{j_1}w_1R_{j_2}\dots R_{j_{n-1}}w_{n-1} = v$ and $a = a_0R_{j_1}a_1R_{j_2}\dots R_{j_{n-1}}$ $a_{n-1} = b$ such that for all i, $M,w_i \models Pre(a_i)$.

Proposition 5. *Let M be a finite epistemic model and let A be a finite serial event model. Let $w \in M$, $a \in A$ and $n = |M| \cdot |A|$.*

The submodel of $M \otimes A$ generated by (w,a) is defined and serial iff $M,w \models \delta^n(a)$.

Proof. First, note that the submodel of $M \otimes A$ generated by (w,a) is defined and serial iff

- (w,a) is defined;
- $R_j(w,a) \neq \emptyset$ for all $j \in G$;
- $R_j(v,b) \neq \emptyset$ for all $(v,b) \in \left(\bigcup_{j\in G} R_j\right)^+ (w,a)$ and for all $j \in G$.

Then we get easily the expected result by Lemma 2 and Lemma 1. Indeed, (w,a) is defined and $R_j(w,a) \neq \emptyset$ for all $j \in G$ amounts to say that $M,w \models \delta^1(a)$. And $R_j(v,b) \neq \emptyset$ for all $(v,b) \in \left(\bigcup_{j\in G} R_j\right)^+ (w,a)$ and for all $j \in G$ amounts to say that for all $v \in M$ such that $w = w_0R_{j_1}w_1R_{j_2}\dots R_{j_n}w_n = v$ such that there are $a = a_0R_{j_1}a_1R_{j_2}\dots R_{j_n}a_n = b$ such that for all $i \in \{0,\dots,n\}$, $M,w_i \models Pre(a_i)$, $M,v \models \bigwedge_{j\in G} \hat{B}_j \bigvee_{c\in R_j(b)} Pre(c)$.

This proposition is coherent with our interpretation of $M,w \models \delta^n(a)$. As we said, intuitively, $M,w \models \delta^n(a)$ means that the submodel of $M \otimes A$ generated by (w,a) is (defined and) serial up to modal depth n. So, if n is larger than the modal depth of the submodel $M \otimes A$ generated by (w,a) (which is the case if $n = |M| \cdot |A|$) then all the worlds accessible from (w,a) are serial. So this generated submodel is indeed serial. Accordingly, this also entails that it should be serial for any given modal depth. That is what the following property expresses.

Proposition 6. *Let M be a finite epistemic model and let A be a finite and serial event model. Let $w \in M, a \in A$ and $n = |M| \cdot |A|$.*
If $M,w \models \delta^n(a)$ then for all $m \geq n$, $M,w \models \delta^m(a)$.

Proof. The proof follows from Lemma 1 and the fact that for all $v \in M$ there are $w_1,\dots,w_{n-1}$ such that $w = w_0R_{j_1}w_1R_{j_2}\dots R_{j_n}w_n = v$ iff there are $w_1,\dots,w_{m-1}$ such that $w = w_0R_{j_1}w_1R_{j_2}\dots R_{j_m}w_m = v$.

Similarly, if a submodel of $M \otimes A$ generated by (w, a) is serial up to a given modal depths d then it should also be serial up to all modal depth smaller than d. The following proposition proves that it is indeed the case.

Proposition 7. *For all event models A and $a \in A$, if $n \geq n'$ then $\models \delta^n(a) \rightarrow \delta^{n'}(a)$.*

Proof. Let A be an event model and $a \in A$. We prove it by induction on n. If $n = 0$ or $n = 1$ then the result trivially holds. Assume it is true for a given $n \geq 1$. Assume $\models \delta^{n+1}(a)$, i.e. $\models \delta^0(a) \wedge \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} \delta^n(b) \wedge \bigwedge_{j \in G} B_j \bigwedge_{b \in R_j(a)} (Pre(b) \rightarrow \delta^n(b))$.

By induction hypothesis, for all $b \in A$, $\models \delta^n(b) \rightarrow \delta^{n-1}(b)$. So

$$\models \left(\delta^0(a) \wedge \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} \delta^n(b) \wedge \bigwedge_{j \in G} B_j \bigwedge_{b \in R_j(a)} (Pre(b) \rightarrow \delta^n(b)) \right) \rightarrow$$

$$\left(\delta^0(a) \wedge \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} \delta^{n-1}(b) \wedge \bigwedge_{j \in G} B_j \bigwedge_{b \in R_j(a)} (Pre(b) \rightarrow \delta^{n-1}(b)) \right).$$

i.e. $\models \delta^{n+1}(a) \rightarrow \delta^n(a)$. So for all $n' \leq n+1$, $\models \delta^{n+1}(a) \rightarrow \delta^{n'}(a)$ by induction hypothesis.

Finally, we can strike some relationship between the seriality conditions for the entire updated model and for the generated submodels of the entire updated model. Indeed, if the entire updated model is serial then all its generated submodels should be serial up to any modal depth:

Proposition 8. *Let M be an epistemic model and A be a serial event model.*

$$\text{if } M \models \mathcal{S}(A) \text{ then for all } n \geq 0 \; M \models \bigwedge_{a \in A} (Pre(a) \rightarrow \delta^n(a)).$$

Proof. By induction on n.

Besides, one can notice that the entire updated model is serial if and only if all its generated submodels are serial. But in fact, because we consider *all* the generated submodels, it suffices that these generated submodels be serial only up to modal depth 1. That is actually the intuition that led to the definition of $\mathcal{S}(A)$.

Proposition 9. *Let M be an epistemic model and let A be a serial event model. Then,*

$$M \models \mathcal{S}(A) \leftrightarrow O\left(\bigvee_{a \in A} Pre(a) \right) \wedge U \bigwedge_{a \in A} \left(Pre(a) \rightarrow \delta^1(a) \right).$$

$O\left(\bigvee_{a \in A} Pre(a) \right)$ expresses that the updated model is defined. The rest of the formula expresses its seriality. Note that $\delta^1(a) = Pre(a) \wedge \bigwedge_{j \in G} \hat{B}_j \bigvee_{b \in R_j(a)} Pre(b)$, so we have rediscovered the definition of $\mathcal{S}(A)$.

5 Conclusion

We have given conditions under which an entire updated model and its generated submodels are serial. We also introduced the notion of crazy formula which are formulas such that after being publicly announced at least one of the agents' beliefs become inconsistent. We could wonder whether other properties are also preserved during an update. It has been shown that most of the relevant ones like reflexivity, transitivity and euclidicity are preserved [3].

The fact that seriality is not preserved in the BMS system means that it should be enriched with some sort of revision mechanisms so that seriality is restored. For example, in Example 3 Ann should revise her beliefs about Bob after the public announcement. To do this, most of the existing approaches resort to a richer framework by introducing plausibility [1,6] or probability [2,4] but none of them tackles the issue directly in its original form by manipulating accessibility relations. This paper is a preliminary step in that direction.

References

1. Aucher, G.: A combined system for update logic and belief revision. In: Barley, M., Kasabov, N. (eds.) PRIMA 2004. LNCS (LNAI), vol. 3371, pp. 1–17. Springer, Heidelberg (2005)
2. Aucher, G.: Interpreting an action from what we perceive and what we expect. Journal of Applied Non-classical Logics 17(1), 9–38 (2007)
3. Baltag, A., Moss, L.: Logic for epistemic program. Synthese 139(2), 165–224 (2004)
4. Baltag, A., Smets, S.: Probabilistic dynamic belief revision. In: van Benthem, J., Ju, S., Veltman, F. (eds.) A Meeting of the Minds: Proceedings of the Workshop on Logic, Rationality and Interaction, London. Computing Series, vol. 8, College Publications (2007)
5. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Computer Science, vol. 53. Cambridge University Press, Cambridge (2001)
6. van Ditmarsch, H.: Prolegomena to dynamic logic for belief revision. Synthese 147, 229–275 (2005)
7. van Ditmarsch, H., Kooi, B.: The secret of my success. Synthese 151, 201–232 (2006)

Propositional Clausal Defeasible Logic

David Billington

School of ICT, Nathan campus, Griffith University,
Brisbane, Queensland 4111, Australia
d.billington@griffith.edu.au

Abstract. Defeasible logics are non-monotonic reasoning systems that have efficient implementations and practical applications. We list several desirable properties and note that each defeasible logic fails to have some of these properties. We define and explain a new defeasible logic, called clausal defeasible logic (CDL), which has all these properties. CDL is easy to implement, consistent, detects loops, terminates, and has a range of deduction algorithms to cater for a range of intuitions.

Keywords: Defeasible logic, Non-monotonic reasoning, Knowledge representation and reasoning, Artificial intelligence.

1 Introduction

Non-monotonic reasoning systems represent and reason with incomplete information where the degree of incompleteness is not quantified. A very simple and natural way to represent such incomplete information is with a defeasible rule of the form "antecedent ⇒ consequent"; with the meaning that provided there is no evidence against the consequent, the antecedent is sufficient evidence for concluding the consequent. Creating such rules is made easier for the knowledge engineer as each rule need only be considered in isolation. The interaction between the rules is the concern of the logic designer.

Reiter's normal defaults [24] have this form, with the meaning that if the antecedent is accepted and the consequent is consistent with our knowledge so far then accept the consequent. Of course the consequent could be consistent with current knowledge and yet there be evidence against the consequent. This results in multiple extensions. However multiple extensions are avoided by interpreting a defeasible rule as "if the antecedent is accepted and all the evidence against the consequent has been nullified then accept the consequent". This interpretation forms the foundation of a family of non-monotonic logics all based on Nute's original defeasible logic [22]. (Different formal definitions of "accepted", "evidence against", and "nullified" have been used by different defeasible logics.)

Unlike other non-monotonic reasoning systems, these defeasible logics use Nute's very simple and natural "defeasible arrow" to represent incomplete information. This simplicity and naturalness is important when explaining an implementation to a client. All defeasible logics have a priority relation on rules. Although preferences between rules can be simulated by more complex rules, this

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 34–47, 2008.

is not so natural or simple. Defeasible logics use classical negation rather than negation-as-failure, and have a type of rule which warns that a conclusion, c, is too risky but does not support the negation of c. Many defeasible logics cater for different intuitions about what should follow from a reasoning situation.

A key feature of these defeasible logics is that they all have efficient easily implementable deduction algorithms (see [8,20,23] for details). Indeed defeasible logics have been used in an expert system, for learning and planning [23], in a robotic dog which plays soccer [9,10], and to improve the accuracy of radio frequency identification [11]. Defeasible logics have been advocated for various applications including modelling regulations and business rules [3], agent negotiations [13], the semantic web [1,4], modelling agents [16], modelling intentions [15], modelling dynamic resource allocation [17], modelling contracts [12], legal reasoning [18], modelling deadlines [14], and modelling dialogue games [25]. Moreover, defeasible theories, describing policies of business activities, can be mined efficiently from appropriate datasets [19].

The unique features and diverse range of practical applications show that defeasible logics are useful and their language is important for knowledge representation and reasoning. Using defeasible logic as the inference engine in an expert system is obvious. But it is less obvious to use defeasible logic to deal with the error-prone output of sensors (possibly in a robot), because this can be done using classical logic. The advantages of using defeasible logic are that the system can be developed incrementally, there are fewer rules, and the rules are simpler [9,10,11].

In this paper we shall define a new defeasible logic, called clausal defeasible logic (CDL), explain how it works, and show why it is needed.

The rest of the paper has the following organisation. Section 2 shows why CDL is needed. Section 3 gives an overview of CDL. The formal definitions of CDL are in Sections 4 and 5. An explanation of the proof algorithms concludes Section 5. An example is considered in Section 6. Section 7 contains results that show CDL is well-behaved. A summary forms Section 8.

2 Why Clausal Defeasible Logic Is Needed

There are three classes of defeasible logic: the twin logics NDL and ADL [21] and their variants; the logic DL93 [5] and its variants; and plausible logic [8] and its later developments. Table 1 compares CDL and representatives from the three classes by noting which properties hold. From the first class we choose NDL and ADL, from the second class we choose DL93, and from the third class we choose the latest plausible logic PL05 [7].

There are two well-informed but different intuitions about what follows from the defeasible theory $Ambig = \{r_1\colon \{\} \Rightarrow b,\ \ r_2\colon \{\} \Rightarrow a,\ \ r_3\colon \{\} \Rightarrow \neg a,\ r_4\colon \{a\} \Rightarrow \neg b\}$. Rule r_1 says there is evidence for b; r_2 says there is evidence for a; r_3 says there is evidence for not a; and r_4 says that a is evidence for not b. All rules have the same reliability or strength. Because there is equal evidence for and against a, we say a is *ambiguous*. The *ambiguity blocking* intuition says that

Table 1. Logics and their Properties

Property \ Logic	**CDL**	**PL05**	**NDL**	**ADL**	**DL93**
Ambiguity blocking	Yes	Yes	Yes	No	Yes
Ambiguity propagating	Yes	Badly	No	Yes	No*
General conflict	Yes	Yes	Yes	Yes	No
Prove disjunctions	Yes	Yes	No	No	No
Team defeat	Yes	Yes	No	No	Yes
Failure-by-looping	Yes	No	Limited	Limited	No
Linear proof hierarchy	Yes	No	Yes	Yes	No*
Terminates	Yes	Yes	No	No	No
Weak decisiveness	Yes	Yes	Yes	No	Yes

* There is a variant of DL93 [2] that has these properties.

since a is not accepted, r_4 cannot be used and so b should be accepted because of r_1. The *ambiguity propagating* intuition says that although r_4 has been weakened by the ambiguity of a, it has not been weakened enough to ignore r_4. So r_1 is evidence for b and r_4 is still evidence against b. Thus b should also be ambiguous, that is the ambiguity of a has been propagated along r_4 to b.

Defeasible logics deal with factual, as well as defeasible, information. A defeasible logic has the *general conflict* property iff the logic recognises that two defeasible rules, r_1 and r_2, conflict whenever the factual information means that the consequents of r_1 and r_2 cannot both hold. For example if $c \vee d$ is a fact then only a logic with the general conflict property will recognise that $\{a\} \Rightarrow \neg c$ and $\{b\} \Rightarrow \neg d$ conflict.

Only plausible logics can *prove disjunctions* of literals.

Suppose there is a team of rules supporting a and a team of rules supporting $\neg a$, and there is a priority between rules. If there is a rule supporting a that is superior to all the rules supporting $\neg a$ then clearly a should be concluded. This is what NDL and ADL do. But more intuitive results can be obtained if the following *team defeat* property is used. If every rule supporting $\neg a$ is inferior to some rule supporting a then conclude a.

To illustrate *failure-by-looping* consider the following example of a positive loop. $\{r_c\colon \{\} \Rightarrow c,\ \ r_{ab}\colon \{a\} \Rightarrow b,\ \ r_{ba}\colon \{b\} \Rightarrow a,\ \ r_{ac}\colon \{a\} \Rightarrow \neg c\}$. Rule r_c is evidence for c, and r_{ac} is evidence against c. But a cannot be proved because r_{ab} and r_{ba} form a positive loop. Hence the evidence against c has been nullified and so we conclude c. This is what NDL and ADL do.

The following is an example of a negative loop. $NegLoop = \{r_a\colon \{\} \Rightarrow a,\ r_b\colon \{\} \Rightarrow b,\ \ r_c\colon \{\} \Rightarrow c,\ \ r_{ab}\colon \{a\} \Rightarrow \neg b,\ \ r_{ba}\colon \{b\} \Rightarrow \neg a,\ \ r_{ac}\colon \{a\} \Rightarrow \neg c\}$. Again r_c is evidence for c, and r_{ac} is evidence against c. But a cannot be proved because r_{ab} and r_{ba} form a negative loop. Hence the evidence against c has been nullified and so we conclude c. No previous defeasible logic does this.

A logic has a *linear proof hierarchy* iff for any two of its proof algorithms, whatever can be proved by one can be proved by the other or vice versa. For NDL and ADL this means that NDL can prove whatever ADL can prove.

Defeasible logics are designed to be implemented on a computer, and so it would be tidy if their deduction algorithms always *terminated*. The algorithms used in DL93, NDL, and ADL do not terminate when trying to prove c in the *NegLoop* example.

To nullify evidence defeasible logics have to be able to disprove formulas. This means that they can demonstrate in a finite number of steps that there is no proof of the formula. (It does not mean that the negation of the formula can be proved.) It would be nice if every formula could be proved or disproved, called decisiveness in [6]. Unfortunately there are examples that show that decisiveness leads to undesirable results [6]. *Weak decisiveness* means that every formula can be proved, or disproved, or the attempted proof gets into a loop. If such a loop is detected then the proof can be terminated, otherwise the deduction algorithm does not terminate.

For each of the first 6 properties in Table 1, a logic which has this property will give more intuitive results than a logic which does not have the property.

Having a linear proof hierarchy is important because then one does not have to decide which proof algorithm to use. Always use the strongest (most reliable) algorithm. If it succeeds then the weaker algorithms will succeed too. If it fails then use the next weakest (less reliable) algorithm. So different degrees of confidence in a proved result can be achieved without using numbers such as probabilities or certainty factors. Ambiguity propagation gives more reliable results, but proves less, than ambiguity blocking. Moreover a range of ambiguity propagating algorithms can be formed giving a range of reliabilities.

Since CDL is based on PL05 we shall consider the flaws of PL05 more closely. The ambiguity propagating algorithm of PL05 is too strong. Its nullifying subalgorithm is too weak, which makes PL05 fail to have a linear proof hierarchy. We define two totally new ambiguity propagating algorithms based on ideas exhibited in a variant of DL93 [2].

PL05 detects all loops, but it only uses this information to terminate loops. NDL and ADL do not really detect loops, they just use some loops in their failure-by-looping method. If all loops were used in a failure-by-looping method then the logic would be decisive; which, as noted above, leads to undesirable results. CDL determines which loops are usable and which are not. Every loop used by NDL and ADL is regarded as usable by CDL. Moreover CDL gets the desired answer for the *NegLoop* example, see Section 6.

3 Overview of Clausal Defeasible Logic (CDL)

CDL reasons with both factual and plausible information, which is represented by strict rules, defeasible rules, warning rules, and a priority relation, $>$, on the rules. All rules have the form "finite-set-of-literals arrow literal".

Strict rules, for example $A \rightarrow l$, represent the aspects of a situation that are certain. If all the literals in A are proved then l can be deduced, no matter what the evidence against l is.

Defeasible rules, for example $A \Rightarrow l$, mean that if all the literals in A are proved and all the evidence against l has been nullified then l can be deduced.

Warning rules, for example $A \rightsquigarrow \neg l$, warn against concluding usually l, but do not support usually $\neg l$. For example, "Sick birds might not fly." is represented by $\{sick(x), bird(x)\} \rightsquigarrow \neg fly(x)$. The idea is that a bird being sick is not sufficient evidence to conclude that it usually does not fly; it is only evidence against the conclusion that it usually flies.

The priority relation, $>$, on the set of non-strict rules allows the representation of preferences among non-strict rules. The priority relation must be acyclic, but does not have to be transitive. Consider the following two rules.

r_1: $\{bird(x)\} \Rightarrow fly(x)$ [Birds usually fly.]
r_2: $\{quail(x)\} \Rightarrow \neg fly(x)$ [Quails usually do not fly.]

Since r_2 is more specific than r_1 we want $r_2 > r_1$, meaning that r_2 is preferred over r_1.

CDL has six proof algorithms. The μ algorithm is monotonic and uses only strict rules. CDL restricted to μ is essentially classical propositional logic. The ambiguity blocking algorithm is denoted by β. There are two ambiguity propagating algorithms denoted by ρ and π, ρ being more reliable than π. Algorithm ρ requires the co-algorithm ι, which only considers supporting evidence for a formula and ignores all contrary evidence. Algorithm π requires the co-algorithm ε, which sees if there is unbeaten evidence for a formula. Let λ be either ρ or π and λ' be its required co-algorithm. Then the more λ' can prove the less λ can prove. So by changing λ' we can make λ more reliable and closer to μ, or less reliable and closer to β.

The task of proving a formula is done by a recursive proof function P. The input to P is the proof algorithm to be used, the formula to be proved, and the background. The background is an initially empty storage bin into which is put all the literals that are currently being proved as P recursively calls itself. The purpose of this background is to detect loops. The output of P is one of the following proof-values $+1$, 0, or -1. The $+1$ means that the formula is proved in a finite number of steps, 0 means that the proof got into a loop which was detected in a finite number of steps, and -1 means that in a finite number of steps it has been demonstrated that there is no proof of the formula and that this demonstration does not get into a loop.

Some proofs use the finite failure of other proofs. Proofs with a proof-value of -1 are always usable. Moreover CDL determines which proofs with a proof-value of 0 are usable and which are not.

4 The Language of Clausal Defeasible Logic (CDL)

CDL uses a countable propositional language with not ($\neg$), and ($\wedge$), and or ($\vee$). The set of all clauses that are resolution-derivable from a set C of clauses is denoted by $Res(C)$. The complement of a formula and the complement of a set of formulas is now defined.

Definition 1. (*complement*, $\sim$).
C1) If a is an atom then $\sim a$ is $\neg a$.
C2) If f is a formula then $\sim\neg f$ is f.
C3) If F is a set of formulas then $\sim F = \{\sim f : f \in F\}$.
C4) If F is a finite set of formulas then $\sim\wedge F$ is $\vee\sim F$; and $\sim\vee F$ is $\wedge\sim F$.

Definition 2. (subclause, proper subclause). A clause $\vee L$ is a subclause of the clause $\vee M$, denoted $\vee L \leq \vee M$, iff $L \subseteq M$. A clause $\vee L$ is a proper subclause of the clause $\vee M$, denoted $\vee L < \vee M$, iff $L \subset M$.

Definition 3. (rule). A *rule*, r, is a triple $(A(r), arrow(r), c(r))$, where $A(r)$ is a finite set of literals called the set of *antecedents* of r, $arrow(r) \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$, and $c(r)$ is a literal called the *consequent* of r.

Strict rules use the *strict arrow*, $\rightarrow$, and are written $A(r) \rightarrow c(r)$.
Defeasible rules use the *defeasible arrow*, $\Rightarrow$, and are written $A(r) \Rightarrow c(r)$.
Warning rules use the *warning arrow*, $\rightsquigarrow$, and are written $A(r) \rightsquigarrow c(r)$.

Definition 4. $(R_s, R_d, R_w, R[l], R[L], Cl(R_s), Ru(C))$. Let R be any set of rules, C be any set of clauses, L be any set of literals, and l be any literal.
$R_s = \{r \in R : r \text{ is strict}\}$. $R[l] = \{r \in R : l = c(r)\}$.
$R_d = \{r \in R : r \text{ is defeasible}\}$. $R[L] = \{r \in R : c(r) \in L\}$.
$R_w = \{r \in R : r \text{ is a warning rule}\}$. $Cl(R_s) = \{\bigvee[\{c(r)\} \cup \sim A(r)] : r \in R_s\}$.
$Ru(C) = \{\sim(L - \{l\}) \rightarrow l : l \in L \text{ and } \vee L \in C\}$.

A strict rule can be converted to a clause, which is what Cl does. Conversely a clause with n literals can be converted to n strict rules, which is what Ru does.

Definition 5. (cyclic, acyclic). A binary relation, $>$, on any set R is *cyclic* iff there exists a sequence, $(r_1, r_2, \ldots, r_n)$ where $n \geq 1$, of elements of R such that $r_1 > r_2 > \ldots > r_n > r_1$. A relation is *acyclic* iff it is not cyclic.

Definition 6. (priority relation, $>$, $R[l; s]$). If R is a set of rules then $>$ is a *priority relation* on R iff $>$ is an acyclic binary relation on $R_d \cup R_w$.
$R[l; s] = \{t \in R[l] : t > s\}$.

We read $t > s$ as t *has a higher priority than* s, or t is *superior* to s. Notice that strict rules are never superior to or inferior to any rule; and $>$ does not have to be transitive.

Let C be a set of clauses. We want to remove from C all the clauses which are empty, or tautologies, or are proper superclauses of other clauses. The result is called the reduct of C, $Red(C)$, and is formally defined below.

Definition 7. ($Red(C)$, reduct). Let C be a set of clauses. $Red(C)$ is the *reduct* of C iff it satisfies RC1, RC2, and RC3 below.
RC1) $Red(C) \subseteq \{\vee L \in C : L \neq \{\}$ and $\vee L$ is not a tautology$\}$.
RC2) If $\{\vee L, \vee M\} \subseteq Red(C)$ then $\vee L$ is not a proper subclause of $\vee M$.
RC3) If $\vee L \in C - Red(C)$ then either .1) $\vee L$ is empty or a tautology;
or .2) a proper subclause of $\vee L$ is in $Red(C)$.

Given a set R of rules, $Pre(R, L)$, the predecessors of L, is the set of all literals that could affect any literal in the set L of literals. $IPre(R, L)$ is the set if immediate predecessors of L.

Definition 8. ($Pre(R, .)$). Suppose R is a set of rules, L is a finite set of literals, l is a literal, and $i \in \mathbb{N}$.
$IPre(R, L) = \bigcup\{A(r) \cup \sim A(r) : r \in R[L \cup \sim L]\}$.
$IPre^0(R, L) = L \cup \sim L$. $\quad IPre^{i+1}(R, L) = IPre(R, IPre^i(R, L))$.
$Pre(R, L) = \bigcup\{IPre^i(R, L) : i \in \mathbb{N}\}$. $\quad Pre(R, l) = Pre(R, \{l\})$.

Definition 9. (clausal defeasible theory, cdt). Let R be a set of rules. The ordered pair $(R, >)$ is called a *clausal defeasible theory* (*cdt*) iff DT1, DT2, and DT3 all hold.
DT1) $R_s = Ru(Red(Res(Cl(R_s))))$.
DT2) $>$ is a priority relation on $R_d \cup R_w$.
DT3) For all literals, l, $Pre(R, l)$ is finite.

The following notation will be needed later.

Definition 10. ($R_s[L, 2]$, $A^*(R_s[l])$). Let $(R, >)$ be a cdt.
$R_s[L, 2] = \{r \in R_s[L] : A(r) \cap \sim L \neq \{\}\}$. $\quad$ If $\{\} \to l \in R_s$ or $\{\} \to \sim l \in R_s$ then $A^*(R_s[l]) = \{A(r) : r \in R_s[l]\}$; else $A^*(R_s[l]) = \{A(r) : r \in R_s[l]\} \cup \{\{l\}\}$.

5 Clausal Defeasible Logic (CDL)

To define the proof algorithms of CDL we defined the proof function P, which is done by using the following nine auxiliary functions: *Plaus* (Plausible), *For*, *Nulld* (Nullified), *Discred* (Discredited), *Dftd* (Defeated), *Disql* (Disqualified), *Evid* (Evidence), *Imp* (Impotent), and *Unwin* (Unwinnable). All these functions depend on the cdt $\Theta = (R, >)$, and have their output in $\{+1, 0, -1\}$.

For non-empty sets max and min have their usual meaning. But we also define $\max\{\} = -1 \quad$ and $\quad \min\{\} = +1$.

The proof algorithms are explained after their formal definition. In the following C is a finite set of clauses, B and L are finite sets of literals, and l is a literal. All the proof algorithms are the same for conjunctions of clauses, disjunctions of literals, and literals in B.

Definition 11. (P for conjunctions, disjunctions, $l \in B$).
Suppose $\lambda \in \{\mu, \rho, \pi, \beta, \varepsilon, \iota\}$.
$\lambda\wedge$) $P(\lambda, \wedge C, B) = \min\{P(\lambda, c, B) : c \in C\}$.
$\lambda\vee 1$) If $\vee L$ is a tautology then $P(\lambda, \vee L, B) = +1$;
$\lambda\vee 2$) else $P(\lambda, \vee L, B) = \max(\{P(\lambda, l, B) : l \in L\} \cup \{P(\lambda, \wedge(A(r) - \sim L), B) : r \in R_s[L, 2]\})$.
$\lambda 1$) If $l \in B$ then $P(\lambda, l, B) = 0$.

Definition 12. (μ and ι proof algorithms continued).
$\mu 2$) If $l \notin B$ then $P(\mu, l, B) = \max\{P(\mu, \wedge A(r), \{l\} \cup B) : r \in R_s[l]\}$.
$\iota 2$) If $l \notin B$ then $P(\iota, l, B) = \max\{P(\iota, \wedge A(r), \{l\} \cup B) : r \in R_s[l] \cup R_d[l]\}$.

Definition 13. (ρ, π, and β proof algorithms continued).
Suppose $\lambda \in \{\rho, \pi, \beta\}$ and define $\rho' = \iota$, $\pi' = \varepsilon$, and $\beta' = \beta$.
$\lambda 2$) If $l \notin B$ then $P(\lambda, l, B) = \max(\{P(\lambda, \wedge A(r), \{l\} \cup B) : r \in R_s[l]\} \cup \{Plaus(\lambda, l, B)\})$.
$\lambda 3$) $Plaus(\lambda, l, B) = \min(\{For(\lambda, l, B)\} \cup \{Nulld(\lambda, l, B, I) : I \in A^*(R_s[\sim l])\})$.
$\lambda 4$) $For(\lambda, l, B) = \max\{P(\lambda, \wedge A(r), \{l\} \cup B) : r \in R_d[l]\}$.
$\lambda 5$) $Nulld(\lambda, l, B, I) = \max\{Discred(\lambda, l, B, q) : q \in I\}$.
$\lambda 6$) $Discred(\lambda, l, B, q) = \min\{Dftd(\lambda, l, B, s) : s \in R[q]\}$.
$\lambda 7$) $Dftd(\lambda, l, B, s) = \max(\{P(\lambda, \wedge A(t), \{l\} \cup B) : t \in R_d[l; s]\} \cup \{Disql(\lambda, l, B, s)\})$.
$\lambda 8$) If $P(\lambda', \wedge A(s), \{l\} \cup B) = 0$ and $l \notin Pre(R, A(s))$ then $Disql(\lambda, l, B, s) = +1$;
$\lambda 9$) else $Disql(\lambda, l, B, s) = -P(\lambda', \wedge A(s), \{l\} \cup B)$.

Definition 14. (ε proof algorithm continued).
$\varepsilon 2$) If $l \notin B$ then $P(\varepsilon, l, B) = \max(\{P(\varepsilon, \wedge A(r), \{l\} \cup B) : r \in R_s[l]\} \cup \{Evid(\varepsilon, l, B, r) : r \in R_d[l]\})$.
$\varepsilon 3$) $Evid(\varepsilon, l, B, r) = \min(\{P(\varepsilon, \wedge A(r), \{l\} \cup B)\} \cup \{Imp(\varepsilon, l, B, r, I) : I \in A^*(R_s[\sim l])\})$.
$\varepsilon 4$) $Imp(\varepsilon, l, B, r, I) = \max\{Unwin(\varepsilon, l, B, r, q) : q \in I\}$.
$\varepsilon 5$) $Unwin(\varepsilon, l, B, r, q) = \min\{Disql(\varepsilon, l, B, s) : s \in R[q; r]\}$.
$\varepsilon 6$) If $P(\pi, \wedge A(s), \{l\} \cup B) = 0$ and $l \notin Pre(R, A(s))$ then $Disql(\varepsilon, l, B, s) = +1$;
$\varepsilon 7$) else $Disql(\varepsilon, l, B, s) = -P(\pi, \wedge A(s), \{l\} \cup B)$.

Definition 15. ($\Theta(\lambda+)$, $\Theta(\lambda-)$, $\Theta(\lambda 0)$). Suppose Θ is a cdt, P is the proof function of Θ, f is a formula, and $\lambda \in \{\mu, \rho, \pi, \beta, \varepsilon, \iota\}$. We define
$\Theta(\lambda+) = \{f : P(\lambda, f, \{\}) = +1\}$, $\Theta(\lambda 0) = \{f : P(\lambda, f, \{\}) = 0\}$, and
$\Theta(\lambda-) = \{f : P(\lambda, f, \{\}) = -1\}$.

We shall now give some insight into the above proof algorithms. Note that min and max behave like quantifiers. If S is a subset of $\{+1, 0, -1\}$ then
$\min(S) = +1$ iff $\forall i \in S(i = +1)$; $\min(S) = -1$ iff $\exists i \in S(i = -1)$;
$\max(S) = +1$ iff $\exists i \in S(i = +1)$; and $\max(S) = -1$ iff $\forall i \in S(i = -1)$.
Suppose $\lambda \in \{\mu, \rho, \pi, \beta, \varepsilon, \iota\}$.

A conjunction of clauses, $\wedge C$, is proved by proving each clause in C. So for $P(\lambda, \wedge C, B)$ to be $+1$ each $P(\lambda, c, B)$ must be $+1$, where $c \in C$. Hence $\lambda\wedge$.

If a disjunction of literals, $\vee L$, is a tautology then we declare it proved. Hence $\lambda\vee 1$. If $\vee L$ is not a tautology then $\vee L$ is proved by either proving at least one literal in L, $\max(\{P(\lambda, l, B) : l \in L\})$, or by generalising the following idea. Suppose $L = \{a, b, c\}$ and r is the strict rule $\{\sim a, d\} \rightarrow c$. Then the clausal form of r is $\vee\{\sim d, a, c\}$. So proving d will show $\vee\{a, c\}$ and hence $\vee L$. Putting these two parts together gives $\lambda\vee 2$.

Proving literals is much more involved. If the literal to be proved, l, is in the background B then we are already in the process of trying to prove l. So we are now in a loop. Hence $P(\lambda, l, B) = 0$ and so $\lambda 1$. So suppose that the literal to be proved, l, is not in the background B.

Since μ uses only strict rules, to prove l we must prove the conjunction of the antecedent of a strict rule whose consequent is l. Hence $\mu 2$.

The ι algorithm only considers evidence that supports l and ignores all evidence against l. So to prove l we must prove the conjunction of the antecedent of any strict or defeasible rule whose consequent is l, $\max\{P(\iota, \wedge A(r), \{l\} \cup B) : r \in R_s[l] \cup R_d[l]\}$. Hence $\iota 2$.

Now consider the ρ, π, and β algorithms. Suppose $\lambda \in \{\rho, \pi, \beta\}$ and recall that $\rho' = \iota$, $\pi' = \varepsilon$, and $\beta' = \beta$. To prove l we must either prove the conjunction of the antecedent of a strict rule whose consequent is l, $\max(\{P(\lambda, \wedge A(r), \{l\} \cup B) : r \in R_s[l]\}$, or do something else, $\{Plaus(\lambda, l, B)\})$. Hence $\lambda 2$. Note that when we prove the antecedent, $\wedge A(r)$, l must be added to the background. If we are not using a strict rule then we need evidence for l, $For(\lambda, l, B)$, and we need to nullify all the evidence against l, $\{Nulld(\lambda, l, B, I) : I \in A^*(R_s[\sim l])\}$. Hence $\lambda 3$. Each I in $A^*(R_s[\sim l])$ is a set of literals which are inconsistent with l. This gives us the general conflict property.

The evidence for l is established by proving the conjunction of the antecedent of a defeasible rule whose consequent is l. Hence $\lambda 4$.

The evidence against l is nullified by, for each I in $A^*(R_s[\sim l])$, finding a literal, q, in I such that every rule, s, whose consequent is q is defeated. Hence $\lambda 5$ and $\lambda 6$. A rule, s, is defeated by either disqualifying it, $Disql(\lambda, l, B, s)$, or by using team defeat. That is by finding a defeasible rule t whose consequent is l (a member of the team for l) and which is superior to s, and then proving the conjunction of the antecedent of t, $\{P(\lambda, \wedge A(t), \{l\} \cup B) : t \in R_d[l; s]\}$. Hence $\lambda 7$. A rule, s, is disqualified by either using the λ' algorithm to disprove its antecedent, $\lambda 9$; or by showing that using λ' to prove its antecedent loops, $P(\lambda', \wedge A(s), \{l\} \cup B) = 0$, and that the loop does not involve l, $l \notin Pre(R, A(s))$. Hence $\lambda 8$.

Finally consider the ε proof algorithm. To prove l we must either prove the conjunction of the antecedent of a strict rule whose consequent is l, $\max(\{P(\varepsilon, \wedge A(r), \{l\} \cup B) : r \in R_s[l]\}$, or find a defeasible rule, r, whose consequent is l, which is sufficient evidence for l, $\{Evid(\varepsilon, l, B, r) : r \in R_d[l]\}$. Hence $\varepsilon 2$. If we are not using a strict rule then we must prove the conjunction of the antecedent of r and make all the evidence against l impotent, $\{Imp(\varepsilon, l, B, r, I) : I \in A^*(R_s[\sim l])\}$. Hence $\varepsilon 3$. The evidence against l is made impotent by, for each I in $A^*(R_s[\sim l])$, finding a literal, q, in I such that every rule, s, whose consequent is q and which is superior to r is disqualified. Hence $\varepsilon 4$ and $\varepsilon 5$. A rule, s, is disqualified by either using π to disprove its antecedent, $\varepsilon 7$; or by showing that using π to prove its antecedent loops, $P(\pi, \wedge A(s), \{l\} \cup B) = 0$, and that the loop does not involve l, $l \notin Pre(R, A(s))$. Hence $\varepsilon 6$.

6 Example

Consider the following example of a negative loop given in Section 2. $R = \{r_a\colon \{\} \Rightarrow a,\ \ r_b\colon \{\} \Rightarrow b,\ \ r_c\colon \{\} \Rightarrow c,\ \ r_{ab}\colon \{a\} \Rightarrow \neg b,\ \ r_{ba}\colon \{b\} \Rightarrow \neg a,\ r_{ac}\colon \{a\} \Rightarrow \neg c\}$ We show that the ambiguity blocking algorithm β proves c, $P(\beta, c, \{\}) = +1$; but that the ambiguity propagating algorithm π disproves c, $P(\pi, c,) = -1$. Define the cdt Θ by $\Theta = (R, >)$, where $>$ is empty. For each literal l, $A^*(R_s[l]) = \{\{l\}\}$. Also $Pre(R, a) = \{a, \neg a, b, \neg b\} = Pre(R, b)$.

Calculation $C1$

1) $P(\beta, c, \{\}) = \mathit{Plaus}(\beta, c, \{\})$, by $\beta 2$
2) $= \min\{\mathit{For}(\beta, c, \{\}), \mathit{Nulld}(\beta, c, \{\}, \{\neg c\})\}$, by $\beta 3$
3) $\mathit{For}(\beta, c, \{\}) = P(\beta, \wedge\{\}, \{c\})$, by $\beta 4$
4) $= \min\{\} = +1$, by $\beta\wedge$.
5) $\therefore P(\beta, c, \{\}) = \mathit{Nulld}(\beta, c, \{\}, \{\neg c\})$, by $4, 3, 2, 1$.
6) $= \mathit{Discred}(\beta, c, \{\}, \neg c)$, by $\beta 5$
7) $= \mathit{Dftd}(\beta, c, \{\}, r_{ac})$, by $\beta 6$
8) $= \mathit{Disql}(\beta, c, \{\}, r_{ac})$, by $\beta 7$
9) $= +1$, by $C1.1(9)$, $c \notin \mathit{Pre}(R, a)$, $\beta 8$

Calculation $C1.1$

1) $P(\beta, a, \{c\}) = \mathit{Plaus}(\beta, a, \{c\})$, by $\beta 2$
2) $= \min\{\mathit{For}(\beta, a, \{c\}), \mathit{Nulld}(\beta, a, \{c\}, \{\neg a\})\}$, by $\beta 3$
3) $\mathit{For}(\beta, a, \{c\}) = P(\beta, \wedge\{\}, \{a, c\})$, by $\beta 4$
4) $= \min\{\} = +1$, by $\beta\wedge$.
5) $\therefore P(\beta, a, \{c\}) = \mathit{Nulld}(\beta, a, \{c\}, \{\neg a\})$, by $4, 3, 2, 1$.
6) $= \mathit{Discred}(\beta, a, \{c\}, \neg a)$, by $\beta 5$
7) $= \mathit{Dftd}(\beta, a, \{c\}, r_{ba})$, by $\beta 6$
8) $= \mathit{Disql}(\beta, a, \{c\}, r_{ba})$, by $\beta 7$
9) $= -0 = 0$, by $C1.1.1(9)$, $a \in \mathit{Pre}(R, b)$, $\beta 9$

Calculation $C1.1.1$

1) $P(\beta, b, \{a, c\}) = \mathit{Plaus}(\beta, b, \{a, c\})$, by $\beta 2$
2) $= \min\{\mathit{For}(\beta, b, \{a, c\}), \mathit{Nulld}(\beta, b, \{a, c\}, \{\neg b\})\}$, by $\beta 3$
3) $\mathit{For}(\beta, b, \{a, c\}) = P(\beta, \wedge\{\}, \{b, a, c\})$, by $\beta 4$
4) $= \min\{\} = +1$, by $\beta\wedge$.
5) $\therefore P(\beta, b, \{a, c\}) = \mathit{Nulld}(\beta, b, \{a, c\}, \{\neg b\})$, by $4, 3, 2, 1$.
6) $= \mathit{Discred}(\beta, b, \{a, c\}, \neg b)$, by $\beta 5$
7) $= \mathit{Dftd}(\beta, b, \{a, c\}, r_{ab})$, by $\beta 6$
8) $= \mathit{Disql}(\beta, b, \{a, c\}, r_{ab})$, by $\beta 7$
9) $= -0 = 0$, by $C1.1.1.1(1)$, $b \in \mathit{Pre}(R, a)$, $\beta 9$

Calculation $C1.1.1.1$

1) $P(\beta, a, \{b, a, c\}) = 0$, by $\beta 1$.

Calculation $C2$

1) $P(\pi, c, \{\}) = \mathit{Plaus}(\pi, c, \{\})$, by $\pi 2$
2) $= \min\{\mathit{For}(\pi, c, \{\}), \mathit{Nulld}(\pi, c, \{\}, \{\neg c\})\}$, by $\pi 3$
3) $\mathit{For}(\pi, c, \{\}) = P(\pi, \wedge\{\}, \{c\})$, by $\pi 4$
4) $= \min\{\} = +1$, by $\pi\wedge$.
5) $\therefore P(\pi, c, \{\}) = \mathit{Nulld}(\pi, c, \{\}, \{\neg c\})$, by $4, 3, 2, 1$.
6) $= \mathit{Discred}(\pi, c, \{\}, \neg c)$, by $\pi 5$
7) $= \mathit{Dftd}(\pi, c, \{\}, r_{ac})$, by $\pi 6$
8) $= \mathit{Disql}(\pi, c, \{\}, r_{ac})$, by $\pi 7$
9) $= - + 1 = -1$, by $C2.1(6)$, $\pi 9$

Calculation $C2.1$

1) $P(\varepsilon, a, \{c\}) = \mathit{Evid}(\varepsilon, a, \{c\}, r_a)$, by $\varepsilon 2$
2) $\quad = \min\{P(\varepsilon, \wedge\{\}, \{a, c\}), \mathit{Imp}(\varepsilon, a, \{c\}, r_a, \{\neg a\})\}$, by $\varepsilon 3$
3) $P(\varepsilon, \wedge\{\}, \{a, c\}) = \min\{\} = +1$, by $\varepsilon\wedge$.
4) $\therefore P(\varepsilon, a, \{c\}) = \mathit{Imp}(\varepsilon, a, \{c\}, r_a, \{\neg a\})$, by $3, 2, 1$.
5) $\quad = \mathit{Unwin}(\varepsilon, a, \{c\}, r_a, \neg a)$, by $\varepsilon 4$
6) $\quad = \min\{\} = +1$, by $\varepsilon 5$

This example shows how π and β differ, and also how the failure-by-looping mechanism works.

7 Results

Our first result shows that the proof function P and its auxiliary functions really are functions, and that termination and weak decisiveness also hold.

Theorem 1. Let $\Theta = (R, >)$ be a cdt. If $\mathit{fn} \in \{P, \mathit{Plaus}, \mathit{For}, \mathit{Nulld}, \mathit{Discred}, \mathit{Dftd}, \mathit{Disql}, \mathit{Evid}, \mathit{Imp}, \mathit{Unwin}\}$ then fn is a function with co-domain $\{+1, 0, -1\}$.

The proof function P can be defined without using the auxiliary functions. This re-phrasing of P has a similar structure to the definitions used in DL93, and enables P to be written as many complicated conventional inference rules.

The importance of having a linear proof hierarchy was explained in Section 2. The next theorem says that CDL has this property, and also the reverse for disproof.

Theorem 2. Suppose Θ is a cdt.
(1) $P(\mu, f, B) \leq P(\rho, f, B) \leq P(\pi, f, B) \leq P(\beta, f, B) \leq P(\varepsilon, f, B) \leq P(\iota, f, B)$.
(2) $\Theta(\mu+) \subseteq \Theta(\rho+) \subseteq \Theta(\pi+) \subseteq \Theta(\beta+) \subseteq \Theta(\varepsilon+) \subseteq \Theta(\iota+)$.
(3) $\Theta(\iota-) \subseteq \Theta(\varepsilon-) \subseteq \Theta(\beta-) \subseteq \Theta(\pi-) \subseteq \Theta(\rho-) \subseteq \Theta(\mu-)$.

The following theorem shows that if the priority relation is empty then $\varepsilon = \iota$ and $\pi = \rho$.

Theorem 3. Suppose $(R, >)$ is a cdt where $>$ is empty.
(1) $P(\varepsilon, f, B) = P(\iota, f, B)$. (2) $P(\pi, f, B) = P(\rho, f, B)$.

Except for ε, if the antecedent of a strict rule is proved then its consequent can also be proved; and so modus ponens or detachment holds for strict rules.

Theorem 4. Suppose $\Theta = (R, >)$ is a cdt, $\lambda \in \{\mu, \rho, \pi, \beta, \iota\}$, and $A \to l$ is in R_s. If $\wedge A \in \Theta(\lambda+)$ then $l \in \Theta(\lambda+)$.

The following example shows that modus ponens need not hold for ε and strict rules. Define $(R, >)$ as follows. $R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$ where r_1 is $\{a, b\} \to l$, r_2 is $\{a, \neg l\} \to \neg b$, r_3 is $\{b, \neg l\} \to \neg a$, r_4 is $\{\} \Rightarrow b$, r_5 is $\{\} \Rightarrow \neg b$, r_6 is $\{\} \Rightarrow a$, r_7 is $\{l\} \Rightarrow \neg a$; and $>$ is defined by $r_7 > r_6$. Then $P(\varepsilon, a, \{\}) = +1$, and $P(\varepsilon, b, \{\}) = +1$, but $P(\varepsilon, l, \{\}) = 0$. So although the antecedent of r_1 is proved its consequent is not. Hence modus ponens fails for ε and r_1.

Definition 16. (Consistent). A set C of clauses is *consistent* iff $\vee\{\} \notin Res(C)$. C is *inconsistent* iff C is not consistent. $\Theta(\lambda+)$ is *consistent* iff the set of clauses in $\Theta(\lambda+)$ is consistent. $\Theta(\lambda+)$ is *inconsistent* iff $\Theta(\lambda+)$ is not consistent.

Our next result says that our proof algorithms are at least as powerful as resolution; in the sense that if two clauses can be proved then their resolvent can also be proved.

Theorem 5. Suppose $\Theta = (R, >)$ is a cdt, and $\lambda \in \{\mu, \rho, \pi, \beta\}$. If $Cl(R_s)$ is consistent then $\Theta(\lambda+)$ is closed under resolution.

Our final result shows that the proof algorithms do not create inconsistencies, and so are trustworthy.

Theorem 6. If $\Theta = (R, >)$ is a cdt, and $\lambda \in \{\mu, \rho, \pi, \beta\}$ then $\Theta(\lambda+) \cup Cl(R_s)$ is consistent iff $Cl(R_s)$ is consistent.

8 Summary

In Section 1 we argued that, among non-monotonic logics, the family of defeasible logics is important for knowledge representation and reasoning. Defeasible logics are powerful enough for a diverse range of practical applications, and yet their language has a unique combination of expressiveness, simplicity, and naturalness.

However all previous defeasible logics have various faults and so give unintuitive answers for some reasoning puzzles. So a defeasible logic that does not suffer from these faults is needed. Clausal defeasible logic (CDL) is such a logic, and yet it inherits all the desirable properties of the family of defeasible logics. This paper defines and explains CDL, gives an example of how to calculate with both the ambiguity blocking and ambiguity propagating algorithms, and shows that CDL is well-behaved and that its deduction algorithms are trustworthy.

References

1. Antoniou, G.: Nonmonotonic rule system on top of ontology layer. In: Bergmann, R. (ed.) Experience Management. LNCS (LNAI), vol. 2432, pp. 394–398. Springer, Heidelberg (2002)
2. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: A Flexible Framework for Defeasible Logics. In: Proc AAAI 2000, pp. 405–410 (2000)
3. Antoniou, G., Billington, D., Maher, M.J.: On the analysis of regulations using defeasible rules. In: Proc. 32nd Hawaii Intl. Conf. on Syst. Sci (HICSS). IEEE Press, Los Alamitos (1999)
4. Bassiliades, N., Antoniou, G., Vlahavas, I.: DR-DEVICE: A defeasible logic system for the Semantic Web. In: Ohlbach, H.J., Schaffert, S. (eds.) PPSWR 2004. LNCS, vol. 3208, pp. 134–148. Springer, Heidelberg (2004)
5. Billington, D.: Defeasible Logic is Stable. J Logic Computation 3(4), 379–400 (1993)

6. Billington, D.: A Plausible Logic which Detects Loops. In: Proc 10th International Workshop on Nonmonotonic Reasoning, pp. 65–71 (2004) ISBN 92-990021-0-X
7. Billington, D.: The Proof Algorithms of Plausible Logic Form a Hierarchy. In: Zhang, S., Jarvis, R. (eds.) AI 2005. LNCS (LNAI), vol. 3809, pp. 796–799. Springer, Heidelberg (2005)
8. Billington, D., Rock, A.: Propositional Plausible Logic: Introduction and Implementation. Studia Logica 67(2), 243–269 (2001)
9. Billington, D., Estivill-Castro, V., Hexel, R., Rock, A.: Non-monotonic Reasoning for Localisation in RoboCup. In: Proc. 2005 Australasian Conference on Robotics and Automation (2005)
http://www.cse.unsw.edu.au/~acra2005/proceedings/papers/billington.pdf
10. Billington, D., Estivill-Castro, V., Hexel, R., Rock, A.: Using Temporal Consistency to Improve Robot Localisation. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 232–244. Springer, Heidelberg (2007)
11. Darcy, P., Stantic, B., Derakhshan, R.: Correcting Stored RFID Data with Non-Monotonic Reasoning. Int. J. of Principles and Apps. of Info. Sci. and Tech (PAIST) 1(1), 65–77 (2007)
12. Governatori, G.: Representing business contracts in RuleML. International Journal of Cooperative Information Systems 14(2-3), 181–216 (2005)
13. Governatori, G., Dumas, M., ter Hofstede, A.H., Oaks, P.: A formal approach to protocols and strategies for (legal) negotiation. In: Proc. 8th International Conference on Artificial Intelligence and Law (ICAIL 2001), pp. 168–177. ACM Press, New York (2001)
14. Governatori, G., Hulstijn, J., Riveret, R., Rotolo, A.: Characterising Deadlines in Temporal Modal Defeasible Logic. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 486–496. Springer, Heidelberg (2007)
15. Governatori, G., Padmanabhan, V.: A defeasible logic of policy-based intention. In: Gedeon, T.D., Fung, L.C.C. (eds.) AI 2003. LNCS (LNAI), vol. 2903, pp. 414–426. Springer, Heidelberg (2003)
16. Governatori, G., Rotolo, A.: Defeasible logic: Agency, intention and obligation. In: Lomuscio, A., Nute, D. (eds.) DEON 2004. LNCS (LNAI), vol. 3065, pp. 114–128. Springer, Heidelberg (2004)
17. Governatori, G., Rotolo, A., Sadiq, S.: A model of dynamic resource allocation in workflow systems. Research & Practice of IT Database Technology 2004 27, 197–206 (2004)
18. Governatori, G., Rotolo, A., Sartor, G.: Temporalised normative positions in defeasible logic. In: 10th Intl. Conf. on AI and Law (ICAIL 2005), pp. 25–34. ACM Press, New York (2005)
19. Johnston, B., Governatori, G.: An algorithm for the induction of defeasible logic theories from databases. Research & Practice of IT Database Technology 2003 17, 75–83 (2003)
20. Maher, M.J., Rock, A., Antoniou, G., Billington, D., Miller, T.: Efficient Defeasible Reasoning Systems. Intl. J. of Artificial Intelligence Tools 10(4), 483–501 (2001)
21. Maier, F., Nute, D.: Ambiguity Propagating Defeasible Logic and the Well-Founded Semantics. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS (LNAI), vol. 4160, pp. 306–318. Springer, Heidelberg (2006)
22. Nute, D.: Defeasible reasoning. In: 20th Hawaii Intl. Conf. on Syst. Sci., pp. 470–477 (1987)

23. Nute, D.: Defeasible Logic. In: Bartenstein, O., Geske, U., Hannebauer, M., Yoshie, O. (eds.) INAP 2001. LNCS (LNAI), vol. 2543, pp. 151–169. Springer, Heidelberg (2003)
24. Reiter, R.: A Logic for Default Reasoning. Artificial Intelligence 13, 81–132 (1980)
25. Thakur, S., Governatori, G., Padmanabhan, V., Lundstrom, J.E.: Dialogue Games in Defeasible Logic. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 497–506. Springer, Heidelberg (2007)

Complexity and Succinctness Issues for Linear-Time Hybrid Logics

Laura Bozzelli and Ruggero Lanotte

Università dell'Insubria, Via Valleggio 11, 22100 - Como, Italy

Abstract. Full linear-time hybrid logic (HL) is a non-elementary and equally expressive extension of standard LTL + past obtained by adding the well-known binder operators $\downarrow$ and $\exists$. We investigate complexity and succinctness issues for HL in terms of the number of variables and nesting depth of binder modalities. First, we present *direct* automata-theoretic decision procedures for satisfiability and model-checking of HL, which require space of exponential height equal to the nesting depth of binder modalities. The proposed algorithms are proved to be asymptotically optimal by providing matching lower bounds. Second, we show that for the one-variable fragment of HL, the considered problems are elementary and, precisely, EXPSPACE-complete. Finally, we show that for all $0 \leq h < k$, there is a succinctness gap between the fragments HL^k and HL^h with binder nesting depth at most k and h, respectively, of exponential height equal to $k - h$.

1 Introduction

Hybrid logics extend modal and temporal logics with features from first-order logic which provide very natural modeling facilities [BS98]. In particular, they provide a type of atomic formulas, called *nominals*, which represent names for states of a model (hence, nominals correspond to constants in first-order logic). Moreover, they contain the *at operator* $@_n$ which gives 'random' access to the state named by n. They may also include the *downarrow binder operator* $\downarrow x$, which assigns the variable name x to the current state, and the *existential binder operator* $\exists x$, which binds x to some state in the model. Applications of hybrid logics range from verification tasks to reasoning about semistructured data [FR06]. Here, we focus on complexity issues for hybrid logics.

Satisfiability of hybrid logics including the binder operator $\downarrow$ or $\exists$ and interpreted on general structures is undecidable, also for small fragments [ABM01, CF05]. For the class of linear structures (based on the frame of the natural numbers with the usual ordering), the problem is instead decidable [FRS03]. However, satisfiability and model checking of full linear-time hybrid logic (HL, for short), an equally-expressive extension of standard LTL + past (PLTL) [Pnu77] with the binders operators $\downarrow$ and $\exists$, are non-elementarily decidable (recall that for LTL and PLTL, these problems are instead PSPACE-complete [SC85, Var88]), and this already holds for the fragment $\mathsf{HL}(\downarrow)$ of HL obtained by disallowing the $\exists$-operator. This is a consequence of the fact that standard first-order logic over words (FO), which is non-elementary [Sto74], can be linearly translated into $\mathsf{HL}(\downarrow)$ [FRS03]. Moreover, by results in [Sto74], there is a non-elementary gap between the succinctness of $\mathsf{HL}(\downarrow)$ and PLTL. Recently, Schwentick et al. [SW07] show that satisfiability

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 48–61, 2008.

of the one-variable fragment $\mathsf{HL}_1(\downarrow)$ of $\mathsf{HL}(\downarrow)$ is elementary and precisely EXPSPACE-complete, while the fragment $\mathsf{HL}_2(\downarrow)$ of $\mathsf{HL}(\downarrow)$ using at most two-variables remains non-elementarily decidable.

Our Contribution. In this paper we further investigate the linear-time hybrid logic HL, and focus on complexity and succinctness issues in terms of the number of variables and the nesting depth of binder modalities. Note that as shown in [FRS03], for the linear-time setting, *nominals* and the *at operator* $@_n$ can be linearly translated into PLTL (without using $\exists$ and $\downarrow$). Thus, they are not considered in this paper. For each $h \geq 1$, let HL^h and $\mathsf{HL}^h(\downarrow)$ be the fragments of HL and $\mathsf{HL}(\downarrow)$, respectively, consisting of formulas with nesting depth of the binder operators at most h, and let h-EXPSPACE be the class of languages which can be decided in space of exponential height h.

First, we present automata-theoretic decision procedures for satisfiability and model checking of HL based on a translation of HL formulas into a subclass of generalized Büchi alternating word automata (AWA). The construction is *direct* and compositional and is based on a characterization of the satisfaction relation for a given formula φ, in terms of sequences of sets associated with φ (which generalize the classical notion of Hintikka-set of LTL) satisfying determined requirements which can be checked by AWA. The proposed translation lead to algorithms which run in space of exponential height equal to the nesting depth of binder modalities. As a consequence for each $h \geq 1$, satisfiability and model checking of HL^h and $\mathsf{HL}^h(\downarrow)$ are in h-EXPSPACE. We show that the proposed algorithms are asymptotically optimal by providing matching lower bounds, and in particular, we show that h-EXPSPACE-hardness already holds for the fragment $\mathsf{HL}^h_2(\downarrow)$ of $\mathsf{HL}^h(\downarrow)$ using at most two variables.

Second, we show that the complexity of satisfiability and model checking for the one-variable fragment HL_1 of HL is elementary and, precisely, EXPSPACE-complete. Note that our result for satisfiability does not follow from EXPSPACE-completeness of the same problem for the one-variable fragment $\mathsf{HL}_1(\downarrow)$ of $\mathsf{HL}(\downarrow)$ [SW07]. In fact, as shown in [FRS03], the $\exists$-operator can be linearly translated into $\mathsf{HL}(\downarrow)$, but the resulting formula contains an additional variable. In particular, HL_1 can be linearly translated into $\mathsf{HL}_2(\downarrow)$ (using two variables), which is already non-elementary. Thus, actually, we do not know whether HL_1 can be translated into $\mathsf{HL}_1(\downarrow)$ with an elementary blow-up.

Finally, we show that for all $0 \leq k < h$, there is a succinctness gap between HL^h and HL^k of exponential height equal to $h - k$.

Remark. Recall that the only known automata-theoretic decision procedures for FO, where the starting point is the work of Büchi [Buc62] on the decidability of MSO over infinite words (and its first-order fragment FO), are not direct and are based on the closure of ω-regular languages under projection and boolean operations. However, complementation for Büchi nondeterministic automata (NWA) is not trivial and the known constructions such as that based on Safra's determinization result [Saf88] are quite complicated. On the other hand, if we use alternating automata, then complementation (by dualization) is easy, but projection, which is trivial for Büchi NWA, is hard and effectively requires translating AWA back to NWA. Thus, the novelty of the proposed automata-theoretic approach for HL, which can be used also for FO (since FO is linearly

translatable in HL) is that like the standard automatic-theoretic approach for LTL, it is based on a direct construction which does not use closure results.

Due to lack of space, many proofs are omitted and can be found in [BL08].

2 Preliminaries

Definition 1. Let $\mathbb{N}$ be the set of natural numbers. For all $n,h \in \mathbb{N}$, let $\mathsf{Tower}(n,h)$ be defined as: $\mathsf{Tower}(n,0) = n$ and $\mathsf{Tower}(n,h+1) = 2^{\mathsf{Tower}(n,h)}$. For each $h \geq 0$, $\exp[h]$ denotes the class of functions $f : \mathbb{N} \to \mathbb{N}$ such that for some constant $c \geq 1$, $f(n) = \mathsf{Tower}(n^c,h)$ for each n. We denote by h-EXPSPACE the class of languages decided by $\exp[h]$-space bounded deterministic Turing machines.

2.1 The Linear-Time Hybrid Logic HL

For a finite alphabet Σ and a finite or infinite word $w = \sigma_0\sigma_1\ldots$ over Σ, $|w|$ denotes the length of w (we set $w = \infty$ if w is infinite). For each $0 \leq i < |w|$, $w(i)$ denotes the i^{th} symbol σ_i of w, w^i denotes the i^{th} suffix of w, i.e. the word $w^i = \sigma_i\sigma_{i+1}\ldots$, and for $0 \leq i \leq j < |w|$, $w[i,j]$ denotes the finite word $w[i,j] = \sigma_i\sigma_{i+1}\ldots\sigma_j$.

Fix a countable set $\{x_1,x_2,\ldots\}$ of (position) variables. The set of HL formulas over a finite set AP of atomic propositions is defined by the following syntax:

$$\varphi := \top \mid p \mid x_h \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}^{dir}\varphi \mid \varphi\,\mathsf{U}^{dir}\varphi \mid \exists x_h.\varphi$$

where $\top$ denotes `true`, $p \in AP$, $dir \in \{+,-\}$, X^+ and U^+ are the future temporal operators "forward next" and "forward until", X^- ("backward next") and U^- ("backward until") are their past counterparts, and $\exists$ is the *existential binder* operator. We also use classical shortcuts: $\mathsf{F}^+\varphi := \top\,\mathsf{U}^+\varphi$ ("forward eventually") and $\mathsf{F}^-\varphi := \top\,\mathsf{U}^-\varphi$ ("backward eventually"), and their duals $\mathsf{G}^+\varphi := \neg\mathsf{F}^+\neg\varphi$ ("forward always") and $\mathsf{G}^-\varphi := \neg\mathsf{F}^-\neg\varphi$ ("backward always"). Moreover, the *downarrow binder* operator $\downarrow$ [Gor96] can be introduced as an abbreviation as follows: $\downarrow x_h.\varphi := \exists x_h.(x_h \wedge \varphi)$.

The notion of *free* variable (w.r.t. the binder modalities) are obvious generalizations from first-order logic. A formula φ is *open* if there is some variable which occurs free in φ. A non-open formula is called *sentence*. HL over AP is interpreted on finite or infinite words w over 2^{AP}. A *valuation* for w is a mapping g assigning to each variable a position $j < |w|$ of w. The satisfaction relation $(w,i,g) \models \varphi$, meaning that φ holds at position i along w w.r.t. the valuation g, is inductively defined as follows (we omit the rules for propositions in AP, boolean connectives, and the past temporal operators):

$$\begin{array}{lll}
(w,i,g) \models x_h & \texttt{iff} & i = g(x_h) \\
(w,i,g) \models \mathsf{X}^+\varphi & \texttt{iff} & i+1 < |w| \text{ and } (w,i+1,g) \models \varphi \\
(w,i,g) \models \varphi_1\,\mathsf{U}^+\varphi_2 & \texttt{iff} & \text{there is } i \leq n < |w|.\,(w,n,g) \models \varphi_2 \text{ and} \\
 & & \qquad \text{for all } i \leq k < n.\,(w,k,g) \models \varphi_1 \\
(w,i,g) \models \exists x_h.\varphi & \texttt{iff} & (w,i,g[x_h \leftarrow m]) \models \varphi \text{ for some } m < |w|
\end{array}$$

where $g[x_h \leftarrow m](x_h) = m$ and $g[x_h \leftarrow m](x_i) = g(x_i)$ for $i \neq h$. Thus, the $\exists x$-operator binds the variable x to some position in the given word, while the $\downarrow x$-operator binds the variable x to the current position. Note that the satisfaction relation depends only

on the values assigned to the variables occurring free in the given formula φ. We write $(w,i) \models \varphi$ to mean that $(w,i,g_0) \models \varphi$, where g_0 maps each variable to position 0.

In the following, unless stated otherwise, a given HL formula is assumed to be a sentence. The size $|\varphi|$ of a HL formula φ is the number of distinct subformulas of φ. Note that the fragment of HL obtained by disallowing variables and the binders operators corresponds to standard LTL + past (PLTL) [Pnu77]. We denote by $\mathsf{HL}(\downarrow)$, the HL fragment given by PLTL + variables + $\downarrow$-operator. W.l.o.g. we assume that if a formula φ uses at most n-variables, these variables are $x_1,\ldots,x_n$, and we write $(w,i,j_1,\ldots,j_n) \models \varphi$ to mean that $(w,i,g) \models \varphi$ for any valuation for w assigning to variable x_h the value j_h for each $1 \leq h \leq n$. For each $k \geq 0$, HL_k (resp. $\mathsf{HL}_k(\downarrow)$) denotes the fragment of HL (resp., $\mathsf{HL}(\downarrow)$) using at most k variables. For a HL (resp., $\mathsf{HL}(\downarrow)$) formula φ, $\mathsf{d}_\exists(\varphi)$ (resp., $\mathsf{d}_\downarrow(\varphi)$) denotes the nesting depth of modality $\exists$ (resp., $\downarrow$) in φ. For all $h,k \geq 0$, HL^h and HL^h_k denote the fragments of HL and HL_k, respectively, where the nesting depth of the $\exists$-operator is at most h. The fragments $\mathsf{HL}^h(\downarrow)$ and $\mathsf{HL}^h_k(\downarrow)$ can be defined similarly. Note that since PLTL and HL are equally expressive [FRS03], the mentioned HL fragments, which extend PLTL, are equally expressive.

Global and Initial Equivalence. Two HL formulas φ_1 and φ_2 are said to be (*globally*) *equivalent* if for each word w and $i < |w|$, $(w,i) \models \varphi_1$ iff $(w,i) \models \varphi_2$. Moreover, φ_1 and φ_2 are said to be *initially equivalent* if for each non-empty word w, $(w,0) \models \varphi_1$ iff $(w,0) \models \varphi_2$. Note that (global) equivalence implies initial equivalence.

Decision problems. A *Kripke structure* over AP is a tuple $\mathcal{K} = \langle S, s_0, \Delta, L\rangle$, where S is a finite set of states, $s_0 \in S$ is an initial state, $\Delta \subseteq S \times S$ is a transition relation that must be total, and $L : S \to 2^{AP}$ maps each state s to the set of propositions that hold in s. A path of $\mathcal{K}$ is an infinite sequence $\pi = s_0 s_1 \ldots$ such that $(s_i, s_{i+1}) \in \Delta$ for each $i \geq 0$.

We are interested in the following decision problems for a given linear-time hybrid logic $\mathfrak{F}$ over AP (such as HL or one of its mentioned fragments), where for a $\mathfrak{F}$-formula φ, $\mathcal{L}(\varphi)$ denotes the set of *infinite* words w over 2^{AP} such that $(w,0) \models \varphi$:

- The *satisfiability problem* is to decide given a formula φ of $\mathfrak{F}$, whether $\mathcal{L}(\varphi) \neq \emptyset$;
- The (*finite-state*) *model checking problem* is to decide given a formula φ of $\mathfrak{F}$ and a Kripke structure $\mathcal{K}$ over AP, whether $\mathcal{K}$ *satisfies* φ, i.e., whether for all paths $\pi = s_0 s_1 \ldots$ of $\mathcal{K}$, condition $(L(s_0)L(s_1)\ldots,0) \models \varphi$ holds.

Note that the $\exists$-operator can be expressed in terms of the $\downarrow$-operator as: $\exists x.\varphi \equiv \downarrow y.\mathsf{E}\downarrow x.\mathsf{E}(y \wedge \varphi)$, where $\mathsf{E}\psi := \mathsf{F}^-(\neg\mathsf{X}^-\top \wedge \mathsf{F}^+\psi)$. The use of an additional variable y seems necessary, and actually, we do not know whether HL_1 can be translated into $\mathsf{HL}_1(\downarrow)$ with an elementary blow-up.

3 Decision Procedures

In this Section, we describe an automata-theoretic approach to solve satisfiability and (finite-state) model checking of HL based on a *direct* translation of HL formulas into a subclass of generalized Büchi alternating word automata. The proposed translation lead to algorithms for the considered problems which run in space of exponential height equal to the nesting depth of the existential binder operator. Moreover, for formulas

containing at least two variables, we show that these algorithms are asymptotically optimal by providing matching lower bounds. Finally, for the fragment HL_1 of HL consisting of formulas with only one variable, we show that the complexity of the considered problems is elementary, and precisely, EXPSPACE-complete. For the upper bound of this last result, we will use the following proposition essentially establishing that nested occurrences of the $\exists$-operator in HL_1 formulas can be avoided at no cost.

Proposition 1. *Given a HL_1 formula φ over AP, one can construct a HL_1^1 formula ψ (without nested occurrences of the $\exists$-operator) over a set of propositions $\widehat{AP} \supseteq AP$ such that $|\psi| = O(|\varphi|)$ and for each infinite word w over 2^{AP}, $(w,0) \models \varphi$ iff there is an infinite word $\widehat{w}$ over $2^{\widehat{AP}}$ such that $(\widehat{w},0) \models \psi$ and for each $i \geq 0$, $\widehat{w}(i) \cap AP = w(i)$.*

Proof. By a trivial readaptation of the construction used in [SW07] to show that satisfiability of $\mathsf{HL}_1(\downarrow)$ can be linearly reduced to satisfiability of its fragment $\mathsf{HL}_1^1(\downarrow)$. □

3.1 Alternating Automata

In this Subsection, we recall the class of alternating (finite-state) automata on infinite words equipped with a generalized Büchi acceptance condition (generalized Büchi AWA), and focus on a subclass of such automata, introduced here for the first time.

For a finite set X, $\mathcal{B}_p(X)$ denotes the set of positive boolean formulas over X built from elements in X using $\vee$ and $\wedge$ (we also allow the formulas `true` and `false`). A subset Y of X *satisfies* $\theta \in \mathcal{B}_p(X)$ iff the truth assignment assigning `true` to the elements in Y and `false` to the elements of $X \setminus Y$ satisfies θ. The set Y *exactly satisfies* θ if Y satisfies θ and every proper subset of Y does not satisfy θ.

A generalized Büchi AWA is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \mathcal{F} \rangle$, where Σ is an input alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\delta : Q \times \Sigma \to \mathcal{B}_p(Q)$ is a transition function, and $\mathcal{F} = \{F_1, \ldots, F_k\}$ is a set of sets of accepting states. For a state q, a q-run of $\mathcal{A}$ over an infinite word $w \in \Sigma^\omega$ is a Q-labeled tree r such that the root is labeled by q and for each node u with label q' (describing a copy of $\mathcal{A}$ in state q' which reads $w(|u|)$, where $|u|$ denotes the distance of node u from the root), there is a (possibly empty) set $H = \{q_1, \ldots, q_n\} \subseteq Q$ exactly satisfying $\delta(q', w(|u|))$ such that u has n children $u_1, \ldots, u_n$, and for each $1 \leq h \leq n$, u_h has label q_h. The run r is *accepting* if for each infinite path $u_0 u_1 \ldots$ in the tree and each accepting component $F \in \mathcal{F}$, there are infinitely many $i \geq 0$ such that the label of u_i is in F. The ω-language of $\mathcal{A}$, $\mathcal{L}(\mathcal{A})$, is the set of $w \in \Sigma^\omega$ such that there is an accepting q_0-run r of $\mathcal{A}$ over w for some $q_0 \in Q_0$.

As we will see in order to capture HL formulas, it suffices to consider a subclass of AWA, we call *AWA with a main path* (MAWA). Formally, a generalized Büchi MAWA is a generalized Büchi AWA $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \mathcal{F} \rangle$ satisfying the following additional conditions. The set of states Q is partitioned into a set Q_m of *main states* and in a set Q_s of *secondary* states. Moreover, $Q_0 \subseteq Q_m$, and for all $\sigma \in \Sigma$, $q_s \in Q_s$, and $q_m \in Q_m$: (i) $\delta(q_s, \sigma)$ does not contain occurrences of main states, and (ii) $\delta(q_m, \sigma)$ is in (positive) disjunctive normal form and there is exactly one main state in each disjunct (hence, a set $Y \subseteq Q$ exactly satisfies $\delta(q_m, \sigma)$ *only if* $Y \cap Q_m$ is a singleton). These requirements ensure that in every run of $\mathcal{A}$, there is exactly one path π (the *main* path) which visits only nodes labeled by main states, and each node that is not visited by π is labeled

by a secondary state. By a readaptation of the standard construction used to convert a Büchi AWA into an equivalent standard Büchi nondeterministic word automaton (Büchi NWA) with a single exponential-time blow-up [MH84], we obtain the following result (for details, see [BL08]).

Theorem 1. *Given a generalized Büchi MAWA $\mathcal{A}$ with set of states $Q = Q_m \cup Q_s$ and acceptance condition $\mathcal{F} = \{F_1, \ldots, F_k\}$, one can construct a Büchi NWA $\mathcal{A}_N$ with number of states $O(k \cdot |Q_m| \cdot 2^{O(k|Q_s|)})$ such that $\mathcal{L}(\mathcal{A}_N) = \mathcal{L}(\mathcal{A})$.*

3.2 Upper Bounds

In this subsection we describe an automata-theoretic algorithm to solve satisfiability and (finite-state) model-checking of HL. First, we give a non-trivial characterization of the satisfaction relation $(w,0) \models \varphi$, for a given formula φ, in terms of sequences of sets associated with φ (which generalize the classical notion of Hintikka-set of LTL) satisfying determined requirements which can be checked by generalized Büchi MAWA. Then, we describe the translation into MAWA based on this characterization.

Fix $n \geq 1$ and an alphabet $\Sigma = 2^{AP}$, and let $[n] = \{1, \ldots, n\}$. In the following, we consider (possibly open) formulas φ in HL_n over AP. A formula ψ is said to be a *first-level subformula of* φ if there is an occurrence of ψ in φ which is *not* in the scope of the $\exists$-operator. The *closure* $\mathsf{cl}(\varphi)$ of φ is the smallest set containing $\top$, the propositions in AP, the variable x_h for each $h \in [n]$, $\mathsf{X}^-\top$, all the *first-level* subformulas of φ, $\mathsf{X}^{dir}(\psi_1 \mathsf{U}^{dir} \psi_2)$ for any *first-level* subformula $\psi_1 \mathsf{U}^{dir} \psi_2$ of φ with $dir \in \{+,-\}$, and the negations of all these formulas (we identify $\neg\neg\psi$ with ψ). Note that $|\mathsf{cl}(\varphi)| = O(|\varphi|)$.

Now, we define by induction on $\mathsf{d}_\exists(\varphi)$ the set $\mathsf{Atoms}(\varphi)$ of *atoms* of φ: $A \in \mathsf{Atoms}(\varphi)$ iff $A \subseteq \mathsf{cl}(\varphi) \cup \bigcup_{h\in[n]}(\{x_h\} \times \{-,0,+\}) \cup \bigcup_{\exists x_h.\psi \in \mathsf{cl}(\varphi)}(\mathsf{Atoms}(\psi) \times \{\psi\} \times \{h\})$, and the following holds (where $dir \in \{+,-\}$):

1. $\top \in A$;
2. for each $\psi \in \mathsf{cl}(\varphi)$, $\psi \in A$ iff $\neg\psi \notin A$;
3. for each $\psi_1 \wedge \psi_2 \in \mathsf{cl}(\varphi)$, $\psi_1 \wedge \psi_2 \in A$ iff $\psi_1, \psi_2 \in A$;
4. for each $\psi_1 \mathsf{U}^{dir} \psi_2 \in \mathsf{cl}(\varphi)$, $\psi_1 \mathsf{U}^{dir} \psi_2 \in A$ iff or $\psi_2 \in A$ or $\psi_1, \mathsf{X}^{dir}(\psi_1 \mathsf{U}^{dir} \psi_2) \in A$;
5. if $\mathsf{X}^-\psi \in A$, then $\mathsf{X}^-\top \in A$;
6. for each $h \in [n]$, $x_h \in A$ iff $(x_h, 0) \in A$;
7. for each $h \in [n]$, A contains exactly one pair of the form $(x_h, dir) \in A$ for some $dir \in \{+,-,0\}$;
8. if $\mathsf{X}^-\top \notin A$, then for each $h \in [n]$, $(x_h, -) \notin A$;
9. if $(B, \psi, h) \in A$, then $B \cap AP = A \cap AP$, ($\mathsf{X}^-\top \in B$ iff $\mathsf{X}^-\top \in A$), and for each $k \in [n]$ with $k \neq h$ and $dir \in \{+,-,0\}$, $(x_k, dir) \in B$ iff $(x_k, dir) \in A$;
10. for each $\exists x_h.\psi \in \mathsf{cl}(\varphi)$, there is $(B, \psi, h) \in A$ such that $x_h \in B$;
11. for each $\exists x_h.\psi \in \mathsf{cl}(\varphi)$, $\exists x_h.\psi \in A$ iff there is $(B, \psi, h) \in A$ with $\psi \in B$.

Intuitively, an atom of φ describes a maximal set of subformulas of φ which can hold at a position i of a word $w \in (2^{AP})^\omega$ w.r.t. a determined valuation $j_1, \ldots, j_n$ of variables $x_1, \ldots, x_n$. In particular, for each $h \in [n]$, the unique pair $(x_h, dir) \in A$ keeps track whether the position j_h referenced by x_h strictly precedes ($dir = -$), strictly follows ($dir = +$), or coincides ($dir = 0$ and $x_h \in A$) with the current position i. Finally, a triple

$(B,\psi,h) \in A$, where $\exists x_h.\psi \in \mathsf{cl}(\varphi)$, describes the set of subformulas of ψ which hold at position i w.r.t. a valuation of variables $x_1,\ldots,x_n$ of the form $j_1,\ldots,j_{h-1},m,j_{h+1},\ldots,j_n$ for some $m \in \mathbb{N}$. Thus, $\exists x_h.\psi$ holds at position i w.r.t. the valuation $j_1,\ldots,j_n$ iff $\psi \in B$ for some $(B,\psi,h) \in A$ (Property 11). Note that Property 10 ensures that there is a triple $(B,\psi,h) \in A$ describing the set of subformulas of ψ which hold at position i w.r.t. the valuation of variables $x_1,\ldots,x_n$ given by $j_1,\ldots,j_{h-1},i,j_{h+1},\ldots,j_n$. The necessity of Property 10 will be clear in the proof of Theorem 2 below. Assuming w.l.o.g. that each $p \in AP$ occurs in φ, and $x_1,\ldots,x_n$ occur in φ, by construction it easily follows that $|\mathsf{Atoms}(\varphi)| = \mathsf{Tower}(O(|\varphi|),\mathsf{d}_\exists(\varphi)+1)$.

Now, we define by induction on $\mathsf{d}_\exists(\varphi)$ the function Succ_φ which maps each atom $A \in \mathsf{Atoms}(\varphi)$ to a subset of $\mathsf{Atoms}(\varphi)$. Intuitively, if A is the atom associated with the current position i of the given word w, then $\mathsf{Succ}_\varphi(A)$ contains the set of atoms associable to the next position $i+1$ (w.r.t. a given valuation of variables $x_1,\ldots,x_n$). Formally, $A' \in \mathsf{Succ}_\varphi(A)$ iff the following holds

(a) for each $\mathsf{X}^+\psi \in \mathsf{cl}(\varphi)$, $\mathsf{X}^+\psi \in A \Leftrightarrow \psi \in A'$;
(b) for each $\mathsf{X}^-\psi \in \mathsf{cl}(\varphi)$, $\mathsf{X}^-\psi \in A' \Leftrightarrow \psi \in A$;
(c) for each $h \in [n]$, $(x_h,-) \in A'$ iff $(x_h,dir) \in A$ for some $dir \in \{0,-\}$;
(d) for each $h \in [n]$, $(x_h,+) \in A$ iff $(x_h,dir) \in A'$ for some $dir \in \{0,+\}$;
(e) for each $(B,\psi,h) \in A$, there is $(B',\psi,h) \in A'$ such that $B' \in \mathsf{Succ}_\psi(B)$;
(f) for each $(B',\psi,h) \in A'$, there is $(B,\psi,h) \in A$ such that $B' \in \mathsf{Succ}_\psi(B)$.

For $A \in \mathsf{Atoms}(\varphi)$, let $\sigma(A) = A \cap AP$, i.e. the set of propositions in AP occurring in A. For an infinite word w over $\Sigma = 2^{AP}$, $i \in \mathbb{N}$, and $\widehat{j} \in \mathbb{N} \cup \{\infty\}$ such that $i \leq \widehat{j}$, a $(i,\widehat{j},\varphi)$-*path over* w is a sequence of atoms of φ, $\pi = A_i,A_{i+1},\ldots$ of length $\widehat{j}-i+1$ satisfying the following: for each $i \leq l \leq \widehat{j}$, $\sigma(A_l) = w(l)$ and if $l < \widehat{j}$, then $A_{l+1} \in \mathsf{Succ}_\varphi(A_l)$.

If $i = 0$ and $\widehat{j} = \infty$, then π is simply called φ-path over w. A φ-path $\pi = A_0,A_1,\ldots$ over $w \in \Sigma^\omega$ is *fair* iff the following is inductively satisfied:

1. For each $\psi_1 \mathsf{U}^+ \psi_2 \in \mathsf{cl}(\varphi)$, there are infinitely many $i \geq 0$ such that either $\psi_2 \in A_i$ or $\neg(\psi_1 \mathsf{U}^+ \psi_2) \in A_i$;
2. There is $K \geq 0$ such that for each $h \in [n]$, $(x_h,-) \in A_K$, and for all $i \geq K$ and $(B,\psi,h) \in A_i$, there is a *fair* ψ-path starting from B over the suffix w^i of w.

Note that the definition of *fair* φ-path ensures the following important requirement whose proof is immediate.

Lemma 1. *If $\pi = A_0,A_1,\ldots,A_i$ is a $(0,i,\varphi)$-path on $w \in \Sigma^\omega$ and $\pi' = A_i,A_{i+1},\ldots$ is a fair φ-path on w^i, then $\pi \cdot \pi' = A_0,\ldots,A_i,A_{i+1},\ldots$ is a* fair φ*-path on w.*

Let $\pi = A_0,A_1,\ldots$ be a φ-path over an infinite word w. By Properties 6 and 7 in def. of atom and Properties (c) and (d) in def. of Succ_φ, it holds that for each $h \in [n]$, the set $P_h = \{j \in \mathbb{N} \mid x_h \in A_j\}$ is either empty or a singleton. We say that π is *good* if for each $h \in [n]$, the set P_h is a singleton. Note that by Property 8 in def. of atom, Property (c) in def. of Succ_φ, and Property 2 in def. of *fair* φ-path, the following holds:

Lemma 2. *If $\pi = A_0,A_1,\ldots,A_i$ is a fair φ-path on w with $\neg\mathsf{X}^-\top \in A_0$, then π is good.*

Now, we prove the main results of this Subsection. First, we need the following lemma.

Lemma 3. *Let $\pi = A_0, A_1, \ldots$ be a fair φ-path over $w \in \Sigma^\omega$ with $\neg \mathsf{X}^- \top \in A_0$. Then, for all $i \geq 0$, $m \geq i$ and $(B, \psi, h) \in A_i$, there is a fair ψ-path $\nu = B_0, B_1, \ldots$ over w such that $\neg \mathsf{X}^- \top \in B_0$, $B_i = B$, $(B_j, \psi, h) \in A_j$ for all $j \leq m$, and for each $k \in [n] \setminus \{h\}$ and $l \geq 0$, $x_k \in B_l$ iff $x_k \in A_l$.*

Proof. Fix $i \geq 0$, $m \geq i$, and $(B, \psi, h) \in A_i$. Let $K \geq 0$ be the constant (depending on π) of Property 2 in def. of fair φ-path, and let $H \geq max\{m, K\}$. Since $(B, \psi, h) \in A_i$, by Properties (e) and (f) in def. of Succ_φ and Properties 6, 7, and 9 in def. of atom, it follows that there is a $(0, H, \psi)$-path $\rho = B_0, B_1, \ldots, B_H$ over w such that $B_i = B$, $\neg \mathsf{X}^- \top \in B_0$, and for each $0 \leq j \leq H$, $(B_j, \psi, h) \in A_j$ and $(x_k \in B_j$ iff $x_k \in A_j)$ for all $k \in [n] \setminus \{h\}$. Since $(B_H, \psi, h) \in A_H$ and $H \geq K$, by Property 2 in def. of fair φ-path, there is a fair ψ-path of the form $\rho' = B_H, B_{H+1}, \ldots$ over the suffix w^H of w. Moreover, for each $k \in [n]$, $(x_k, -) \in A_K$. By Property (c) in def. of Succ_φ and Properties 6, 7, and 9 in def. of atom, we obtain that $x_k \notin B_j$ and $x_k \notin A_j$ for all $j \geq H$ and $k \in [n] \setminus \{h\}$. Thus, by Lemma 1 (which holds for any formula ψ in HL_n) it follows that $\rho \cdot \rho' = B_0, \ldots, B_H, B_{H+1}, \ldots$ is a fair ψ-path over w satisfying the statement of the lemma. □

Theorem 2 (Correctness). *Let $\pi = A_0, A_1, \ldots$ be a fair φ-path on $w \in \Sigma^\omega$ such that $\neg \mathsf{X}^- \top \in A_0$, and for each $h \in [n]$, let j_h be the unique index such that $x_h \in A_{j_h}$. Then, for each $i \geq 0$ and $\psi \in \mathsf{cl}(\varphi)$, $(w, i, j_1, \ldots, j_n) \models \psi \Leftrightarrow \psi \in A_i$.*

Proof. By induction on $\mathsf{d}_\exists(\varphi)$. The base step ($\mathsf{d}_\exists(\varphi) = 0$) and the induction step ($\mathsf{d}_\exists(\varphi) > 0$) are similar, and we focus on the induction step. Thus, we can assume that the theorem holds for each formula θ such that $\exists x_h.\theta \in \mathsf{cl}(\varphi)$ for some $h \in [n]$ (note that if $\mathsf{d}_\exists(\varphi) = 0$, there is no such formula). Fix $i \geq 0$ and $\psi \in \mathsf{cl}(\varphi)$. By a nested induction on the structure of ψ, we show that $(w, i, j_1, \ldots, j_n) \models \psi \Leftrightarrow \psi \in A_i$. The cases where ψ is a proposition in AP, or ψ has a PLTL operator at its root are managed in a standard way (for details, see [BL08]). For the remaining cases, we proceed as follows:

Case $\psi = x_h$ with $h \in [n]$. $(w, i, j_1, \ldots, j_n) \models x_h \Leftrightarrow i = j_h \Leftrightarrow$ (by def. of j_h) $x_h \in A_i$.

Case $\psi = \exists x_h.\psi_1$ with $h \in [n]$. First, we show the direct implication $(w, i, j_1, \ldots, j_n) \models \psi \Rightarrow \psi \in A_i$. Let $(w, i, j_1, \ldots, j_n) \models \psi$. Then, $(w, i, j_1, \ldots, j_{h-1}, l, j_{h+1}, \ldots, j_n) \models \psi_1$ for some $l \in \mathbb{N}$. By Property 10 in def. of atom there is $(B, \psi_1, h) \in A_l$ such that $x_h \in B$. Let $m \geq \{i, l\}$. Since $\neg \mathsf{X}^- \top \in A_0$, by Lemma 3 there is a fair ψ_1-path $\rho = B_0, B_1, \ldots$ over w such that $B_l = B$ (hence, $x_h \in B_l$), $\neg \mathsf{X}^- \top \in B_0$, $(B_j, \psi, h) \in A_j$ for each $j \leq m$ (hence, $(B_i, \psi, h) \in A_i$), and for each $k \in [n] \setminus \{h\}$, $x_k \in B_{j_k}$. Since the theorem holds for ψ_1 and $(w, i, j_1, \ldots, j_{h-1}, l, j_{h+1}, \ldots, j_n) \models \psi_1$, it follows that $\psi_1 \in B_i$. Since $(B_i, \psi_1, h) \in A_i$, by Property 11 in def. of atom we obtain that $\psi \in A_i$.

For the converse implication, let $\psi \in A_i$. By Property 11 in def. of atom there is $(B, \psi_1, h) \in A_i$ with $\psi_1 \in B$. Since $\neg \mathsf{X}^- \top \in A_0$, by Lemma 3 there is a fair ψ_1-path $\rho = B_0, B_1, \ldots$ over w such that $B_i = B$, $\neg \mathsf{X}^- \top \in B_0$, and for each $k \in [n] \setminus \{h\}$, $x_k \in B_{j_k}$. Let $l \in \mathbb{N}$ be the unique index such that $x_h \in B_l$. Since the theorem holds for ψ_1 and $\psi_1 \in B_i$, we obtain that $(w, i, j_1, \ldots, j_{h-1}, l, j_{h+1}, \ldots, j_n) \models \psi_1$, hence $(w, i, j_1, \ldots, j_n) \models \psi$. □

Moreover, we can show the following result (a proof is in [BL08]).

Theorem 3 (Completeness). *Let $w \in \Sigma^\omega$ and $j_1, \ldots, j_n \in \mathbb{N}$. Then, there is a fair φ-path $\pi = A_0, A_1, \ldots$ over w such that $\neg \mathsf{X}^- \top \in A_0$ and for each $k \in [n]$, $x_k \in A_{j_k}$.*

By Theorems 2 and 3 we obtain the following characterization of $(w,0) \models \varphi$.

Corollary 1. *For each word $w \in \Sigma^\omega$, $(w,0) \models \varphi$ iff there is a fair φ-path $\pi = A_0, A_1, \ldots$ over w such that $\varphi, \neg \mathsf{X}^- \top, x_h \in A_0$ for each $h \in [n]$.*

Translation into MAWA Now, we illustrate the translation of HL formulas into generalized Büchi MAWA based on the result of Corollary 1.

Theorem 4. *For a HL formula φ over AP, one can build a generalized Büchi MAWA $\mathcal{A}_\varphi$ over 2^{AP} with states $Q_m \cup Q_s$ and $O(|\varphi|)$ Büchi components such that $\mathcal{L}(\mathcal{A}_\varphi) = \mathcal{L}(\varphi)$, $|Q_m| = \mathsf{Tower}(O(|\varphi|), \mathsf{d}_\exists(\varphi)+1)$, and $|Q_s| = \mathsf{Tower}(O(|\varphi|), \mathsf{d}_\exists(\varphi))$.*

Proof. Let $x_1, \ldots, x_n$ be the variables occurring in φ. We construct a generalized Büchi MAWA $\mathcal{A}_\varphi$ of the desired size with set of main states containing $\mathsf{Atoms}(\varphi)$ and set of initial states given by $\{A \in \mathsf{Atoms}(\varphi) \mid \varphi, \neg \mathsf{X}^- \top, x_h \in A \text{ for each } h \in [n]\}$ such that for each $A \in \mathsf{Atoms}(\varphi)$ and $w \in \Sigma^\omega$, $\mathcal{A}_\varphi$ has an accepting A-run over w iff there is a fair φ-path over w starting from A. Hence, the result follows from Corollary 1. The construction is given by induction on $\mathsf{d}_\exists(\varphi)$. Thus, we can assume that for each $\exists x_h.\psi \in \mathsf{cl}(\varphi)$, one can construct the MAWA $\mathcal{A}_\psi$ associated with ψ. Here, we informally describe the construction (the formal definition is in [BL08]).

Assume that $\mathcal{A}_\varphi$ starts the computation over an input w in a main state $A \in \mathsf{Atoms}(\varphi)$. Then, $\mathcal{A}_\varphi$ guesses a φ-path $\pi = A_0, A_1, \ldots$ over w (with $A_0 = A$) by simulating it along the main path of the run tree. In order to check that π satisfies Property 2 in def. of fair φ-path, $\mathcal{A}_\varphi$ guesses a point along the main path (the constant K in Property 2), checks that $(x_h, -)$ is in the current guessed atom for each $h \in [n]$, and from this instant forward, for each triple (B, ψ, h) in the current guessed atom, $\mathcal{A}_\varphi$ starts an additional copy of the MAWA $\mathcal{A}_\psi$ in state B (which represents a secondary $\mathcal{A}_\varphi$-state). Finally, the acceptance condition of $\mathcal{A}_\varphi$ extends the acceptance conditions of the MAWAs $\mathcal{A}_\psi$ with additional sets used to check that the infinite sequence of states visited by the main copy of $\mathcal{A}_\varphi$ (corresponding to the simulated φ-path) satisfies Property 1 in def. of fair φ-path. □

Corollary 2. *Given a HL formula φ over AP, one can build a Büchi NWA $\mathcal{A}_{N,\varphi}$ over 2^{AP} of size $\mathsf{Tower}(O(|\varphi|), \mathsf{d}_\exists(\varphi)+1)$ s.t. $\mathcal{L}(\mathcal{A}_{N,\varphi}) = \mathcal{L}(\varphi)$. Moreover, if φ is an HL_1 formula, then one can build an equivalent Büchi NWA of size doubly exponential in $|\varphi|$.*

Proof. The first result follows from Theorems 1 and 4. For the second one, note that for $\widehat{AP} \supset AP$ and a Büchi NWA $\mathcal{A}_N$ over $2^{\widehat{AP}}$, $\mathcal{L}(\mathcal{A}_N)$ can be seen as a language on $2^{AP} \times 2^{\widehat{AP} \setminus AP}$. Since one can build a Büchi NWA of the same size as $\mathcal{A}_N$ accepting the projection of $\mathcal{L}(\mathcal{A}_N)$ on 2^{AP}, the result follows from Proposition 1 and Theorems 1 and 4. □

By Corollary 2, the model checking problem for an HL formula φ is reduced to emptiness of the Büchi NWA $\mathcal{A}_{\neg\varphi,\mathcal{K}}$ obtained as the synchronous product of the Büchi NWA $\mathcal{A}_\mathcal{K}$ (where each state is accepting) corresponding to the given Kripke structure $\mathcal{K}$ and the Büchi NWA $\mathcal{A}_{N,\neg\varphi}$ associated with the *negation* of formula φ. Since nonemptiness of Büchi NWA is in NLOGSPACE, by Corollary 2 we obtain the following result.

Theorem 5 (Upper bounds). *Satisfiability and model checking of HL_1 and HL^h (for any $h \geq 1$) are in* EXPSPACE *and h-*EXPSPACE*, respectively.*

3.3 Lower Bounds

In this Subsection we show that for each $h \geq 1$, satisfiability and model-checking of $\mathsf{HL}_2^h(\downarrow)$ are h-EXPSPACE-hard by a reduction from the word problem for $\exp[h]$-space bounded deterministic Turing Machines. In the following, w.l.o.g. we assume that the considered HL formulas φ over a finite set of propositions AP are interpreted on words over 2^{AP} where each symbol is a singleton (these words can be seen as words over AP).

Fix $n \geq 1$, a finite alphabet $\Sigma \cup \{0,1\}$, and a countable set $\{\$_1, \$_2, \ldots\}$ of symbols non in $\Sigma \cup \{0,1\}$. First, for each $h \geq 1$, we define by induction on h an encoding of the integers in $[0, \mathsf{Tower}(n,h)-1]$ by finite words, called (h,n)-*codes*, over $\{\$_1, \ldots, \$_h, 0, 1\}$ of the form $\$_h w \$_h$, where w does not contain occurrences of $\$_h$.

Base Step. $h = 1$. A $(1,n)$-*block* over Σ is a finite word w over $\{\$_1, 0, 1\} \cup \Sigma$ having the form $w = \$_1 \sigma b_1 \ldots b_n \$_1$, where $\sigma \in \Sigma \cup \{0,1\}$ and $b_1, \ldots, b_n \in \{0,1\}$. The *block-content* $\mathsf{CON}(w)$ of w is σ, and the *block-number* $\mathsf{NUM}(w)$ of w is the natural number in $[0, \mathsf{Tower}(n,1)-1]$ (recall that $\mathsf{Tower}(n,1) = 2^n$) whose binary code is $b_1 \ldots b_n$. An $(1,n)$-*code* is a $(1,n)$-block w such that $\mathsf{CON}(w) \in \{0,1\}$.

Induction Step. let $h \geq 1$. A $(h+1,n)$-*block* on Σ is a word w on $\{\$_1, \ldots, \$_{h+1}, 0, 1\} \cup \Sigma$ of the form $\$_{h+1} \sigma \$_h w_1 \$_h w_2 \$_h \ldots \$_h w_K \$_h \$_{h+1}$, where $\sigma \in \{0,1\} \cup \Sigma$, $K = \mathsf{Tower}(n,h)$ and for each $1 \leq i \leq K$, $\$_h w_i \$_h$ is a (h,n)-*code* such that $\mathsf{NUM}(\$_h w_i \$_h) = i-1$. The *block-content* $\mathsf{CON}(w)$ of w is the symbol σ, and the *block-number* $\mathsf{NUM}(w)$ of w is the natural number in $[0, \mathsf{Tower}(n,h+1)-1]$ whose binary code is given by $\mathsf{CON}(\$_h w_1 \$_h) \ldots \mathsf{CON}(\$_h w_K \$_h)$. A $(h+1,n)$-*code* is a $(h+1,n)$-block w such that $\mathsf{CON}(w) \in \{0,1\}$.

For each $h \geq 1$, a (h,n)-configuration over Σ is a finite word w of the form $w = \$_{h+1} \$_h w_1 \$_h w_2 \$_h \ldots \$_h w_K \$_h \$_{h+1}$, where $K = \mathsf{Tower}(n,h)$ and for any $1 \leq i \leq K$, $\$_h w_i \$_h$ is a (h,n)-*block* such that $\mathsf{NUM}(\$_h w_i \$_h) = i-1$ and $\mathsf{CON}(\$_h w_i \$_h) \in \Sigma$. As we will see, (h,n)-configurations are used to encode the configurations reachable by $\exp[h]$-space bounded deterministic Turing machines on inputs of size n.

We will use the following non-trivial technical result, whose proof is in [BL08], where for each $h \geq 1$, $\mathsf{Parity}(h) := 1$ if h is odd, and $\mathsf{Parity}(h) := 2$ otherwise.

Proposition 2. *For each $h \geq 1$, we can construct two $\mathsf{HL}_2^{h-1}(\downarrow)$ formulas ψ_h^{conf} and $\psi_h^{=}$ over $\{\$_1, \ldots, \$_{h+1}, 0, 1\} \cup \Sigma$ of sizes bounded by $O(n^3 \cdot h \cdot |\Sigma|)$ such that $\psi_h^{=}$ is open and for $w \in \Sigma^\omega$ and $i \geq 0$, we have*

- *for all j_1, j_2, $(w, i, j_1, j_2) \models \psi_h^{conf}$ iff w^i has a prefix that is a (h,n)-configuration;*
- *if there is $j > i$ such that $w[i,j] = \$_h w_1 \$_h w' \$_h w_2 \$_h$, where $\$_h w_1 \$_h$ and $\$_h w_2 \$_h$ are (h,n)-blocks over Σ, then for each $m \geq 0$,*
  - ***Case** $\mathsf{Parity}(h) = 1$: $(w,i,j,m) \models \psi_h^{=}$ iff $\mathsf{NUM}(\$_h w_1 \$_h) = \mathsf{NUM}(\$_h w_2 \$_h)$;*
  - ***Case** $\mathsf{Parity}(h) = 2$: $(w,j,m,i) \models \psi_h^{=}$ iff $\mathsf{NUM}(\$_h w_1 \$_h) = \mathsf{NUM}(\$_h w_2 \$_h)$.*

Theorem 6. *For each $h \geq 1$, the satisfiability and model checking problems for $\mathsf{HL}_2^h(\downarrow)$ are both h-EXPSPACE-hard.*

Proof. Fix $h \geq 1$. First, we consider the satisfiability problem for $\mathsf{HL}_2^h(\downarrow)$. Let $\mathcal{M} = \langle A, Q, q_0, \delta, F \rangle$ be an $\exp[h]$-space bounded Turing Machine (TM, for short) without halting configurations, and let $c \geq 1$ be a constant such that for each $\alpha \in A^*$, the space

needed by $\mathcal{M}$ on input α is bounded by $\mathsf{Tower}(|\alpha|^c, h)$. For $\alpha \in A^*$, we construct a $\mathsf{HL}_2^h(\downarrow)$ formula $\varphi_{\mathcal{M},\alpha}$ of size *polynomial* in $n = |\alpha|^c$ and in the size of $\mathcal{M}$, such that $\mathcal{M}$ accepts α *iff* $\varphi_{\mathcal{M},\alpha}$ is satisfiable.

Note that any reachable configuration of $\mathcal{M}$ over α can be seen as a word $\alpha_1 \cdot (q,a) \cdot \alpha_2$ in $A^* \cdot (Q \times A) \cdot A^*$ of length $\mathsf{Tower}(n,h)$, where $\alpha_1 \cdot a \cdot \alpha_2$ denotes the tape content, q the current state, and the reading head is at position $|\alpha_1| + 1$. If $\alpha = a_1 \dots a_r$ (where $r = |\alpha|$), then the initial configuration is given by $(q_0, a_1)a_2 \dots a_r \underbrace{\#\#\dots\#}_{\mathsf{Tower}(n,h)-r}$, where # is the blank symbol. Let $C = u_1 \dots u_{\mathsf{Tower}(n,h)}$ be a TM configuration. For $1 \le i \le \mathsf{Tower}(n,h)$, the value u_i' of the i^{th} cell of the $\mathcal{M}$-successor of C is completely determined by the values u_{i-1}, u_i and u_{i+1} (taking u_{i+1} for $i = \mathsf{Tower}(n,h)$ and u_{i-1} for $i = 1$ to be some special symbol). Let $next_{\mathcal{M}}(u_{i-1}, u_i, u_{i+1})$ be our expectation for u_i' (this function can be trivially obtained from the transition function δ of $\mathcal{M}$).

Let $\Sigma = A \cup (Q \times A)$. The *code* of a TM configuration $C = u_1 \dots u_{\mathsf{Tower}(n,h)}$ is the (h,n)-configuration over Σ given by $\$_{h+1}\$_h w_1 \$_h \dots \$_h w_{\mathsf{Tower}(n,h)} \$_h \$_{h+1}$, where for each $1 \le i \le \mathsf{Tower}(n,h)$, $\mathsf{CON}(\$_h w_i \$_h) = u_i$. A non-empty finite sequence of TM configurations $\wp = C_0 C_1 \dots C_m$ is encoded by the infinite word over $\Sigma \cup \{0,1,acc,\$_1,\dots\$_{h+1}\}$, called *sequence-code*, given by $w_\wp = \$_{h+1} w_{C_0} \$_{h+1} \dots \$_{h+1} w_{C_m} \$_{h+1} (acc)^\omega$, where for each $0 \le i \le m$, $\$_{h+1} w_{C_i} \$_{h+1}$ is the code of configuration C_i. The sequence-code $w_\wp$ is *good* iff C_0 is the initial TM configuration over α, C_m is an accepting TM configuration (i.e., the associated state is in F), and $\wp$ is faithful to the evolution of $\mathcal{M}$. Thus, $\mathcal{M}$ accepts α iff there is a good sequence-code.

Now, we build a $\mathsf{HL}_2(\downarrow)$ formula $\varphi_{\mathcal{M},\alpha}$ which is (initially) satisfied by a word w iff w is a good sequence-code. Hence, $\mathcal{M}$ accepts α iff $\varphi_{\mathcal{M},\alpha}$ is satisfiable. Formula $\varphi_{\mathcal{M},\alpha}$ uses the formulas $\psi_h^=$ and ψ_h^{conf} of Proposition 2 (for fixed n and Σ), and is given by

$$\varphi_{\mathcal{M},\alpha} = \varphi_{SC} \wedge \varphi_{first} \wedge \varphi_{acc} \wedge \varphi_\delta$$

where: (1) φ_{SC} uses ψ_h^{conf} and checks that the given word is a sequence-code of some sequence of TM configurations $\wp = C_0, \dots, C_m$, (2) φ_{first} is a PLTL formula checking that C_0 is the initial configuration, (3) φ_{acc} is a PLTL formula checking that C_m is an accepting configuration, and (4) φ_δ uses $\psi_h^=$ and checks that $\wp$ is faithful to the evolution of $\mathcal{M}$. The construction of φ_{first} and φ_{acc} is simple. Thus, we focus on φ_{SC} and φ_δ.

$$\varphi_{SC} = \$_{h+1} \wedge (\mathsf{X}^+ \neg acc) \wedge ((\$_{h+1} \rightarrow \underline{\psi_h^{conf}})\ \mathsf{U}^+\ (\$_{h+1} \wedge \mathsf{G}^+ \mathsf{X}^+ acc))$$

(recall that $(w,i,j_1,j_2) \models \psi_h^{conf}$ iff w^i has a prefix that is a (h,n)-configuration over Σ).

Finally, we define formula φ_δ, which uses $\psi_h^=$. Here, we assume that $\mathsf{Parity}(h) = 1$ (the other case being similar). Recall that if $\mathsf{Parity}(h) = 1$, then for each subword $w[i,j]$ of the given word w and $m \ge 0$ such that $w[i,j] = bl \cdot w' \cdot bl'$, where bl and bl' are (h,n)-blocks, then $(w,i,j,m) \models \psi_h^=$ iff $\mathsf{NUM}(bl) = \mathsf{NUM}(bl')$.

For a sequence-code w, we have to require that for each subword $\$_{h+1} w_1 \$_{h+1} w_2 \$_{h+1}$, where $\$_{h+1} w_1 \$_{h+1}$ and $\$_{h+1} w_2 \$_{h+1}$ encode two TM configurations C_1 and C_2, C_2 is the TM successor of C_1, i.e., for each (h,n)-block bl' of $\$_{h+1} w_2 \$_{h+1}$, the block-content u' of bl' satisfies $u' = next_{\mathcal{M}}(u_p, u, u_s)$, where u is the block-content of the (h,n)-block bl of $\$_{h+1} w_1 \$_{h+1}$ *having the same block-number as* bl', and u_p (resp., u_s) is the block-content of the (h,n)-block of $\$_{h+1} w_1 \$_{h+1}$ — if any — that precedes (resp., follows)

bl. We define only the formula which encodes the case in which bl' is a non-extremal (h,n)-block. The other cases can be handled similarly. Such a formula is defined as follows, where for $u' \in \Sigma$, $H(u')$ is the set of triples $(u_p,u,u_s) \in [\Sigma]^3$ such that $u' = next_{\mathcal{M}}(u_p,u,u_s)$:

$$\bigwedge_{u' \in \Sigma} \mathsf{G}^+\big(\{u' \wedge (\neg \$_h \mathsf{U}^+(\$_h \wedge \mathsf{X}^+ \neg \$_{h+1})) \wedge (\neg \mathsf{X}^{-3} \$_{h+1}) \wedge \mathsf{F}^-(\$_{h+1} \wedge \mathsf{X}^- \top)\} \longrightarrow \{\neg \$_h \mathsf{U}^+(\$_h \wedge \bigvee_{(u_p,u,u_s) \in H(u')} \downarrow x_1 . \psi_\delta^{u_p,u,u_s})\}\big)$$

$$\psi_\delta^{u_p,u,u_s} := \mathsf{F}^-\Big(u_p \wedge \{\neg \$_h \mathsf{U}^+(\$_h \wedge \underline{\psi_h^=} \wedge \mathsf{X}^+(u \wedge (\neg \$_h \mathsf{U}^+(\$_h \wedge \mathsf{X}^+ u_s))))\} \wedge \{\neg \$_{h+1} \mathsf{U}^+(\$_{h+1} \wedge \mathsf{X}^+(\neg \$_{h+1} \mathsf{U}^+ x_1))\}\Big)$$

By Proposition 2 it follows that $\varphi_{\mathcal{M},\alpha}$ is a $\mathsf{HL}_2^h(\downarrow)$ formula of size polynomial in the size of α and $\mathcal{M}$. Model-checking for $\mathsf{HL}_2^h(\downarrow)$ is also h-EXPSPACE-hard since (1) non-satisfiability is linearly reducible to validity (note that $\mathsf{HL}_2^h(\downarrow)$ is closed under negation), and (2) validity is linearly reducible to model-checking [DS02]. □

Since satisfiability and model checking of $\mathsf{HL}_1(\downarrow)$ are EXPSPACE-hard [SW07], by Theorems 5 and 6, we obtain the following corollary.

Corollary 3. *Satisfiability and model checking of HL_1 are* EXPSPACE*-complete. Moreover, for all $h \geq 1$ and $k \geq 2$, satisfiability and model checking of HL_k^h and $\mathsf{HL}_k^h(\downarrow)$ are h-*EXPSPACE*-complete.*

4 Succinctness Issues

In this Section, we show that for all $h > k \geq 0$, there is a succinctness gap between HL^h and HL^k of exponential height equal to $h-k$. Actually, we show a stronger result: for each $h \geq 1$, there is an alphabet Σ_h and a family of $\mathsf{HL}_2^h(\downarrow)$ formulas $(\varphi_{h,n})_{n \geq 1}$ over Σ_h such that for each $n \geq 1$, $\varphi_{h,n}$ has size polynomial in n and each initially equivalent HL^k formula, for $k < h$, has size at least $\mathsf{Tower}(\Omega(n), h-k)$.

We use the encoding defined in Subsection 3.3. For all $n,h \geq 1$, a *finite* (h,n)*-good word* w is a finite word over the alphabet $\{0,1,\$_1,\ldots,\$_{h+1}\}$ having the form $w = \$_{h+1} w_1 \$_{h+1} \ldots \$_{h+1} w_m \$_{h+1}$ such that $m > 1$, for each $1 \leq i \leq m$, $\$_{h+1} w_i \$_{h+1}$ is a $(h+1,n)$-code, and the following holds:

Case $\mathsf{Parity}(h) = 1$**.** There is $1 < i \leq m$ s.t. $\mathsf{NUM}(\$_{h+1} w_1 \$_{h+1}) = \mathsf{NUM}(\$_{h+1} w_i \$_{h+1})$;

Case $\mathsf{Parity}(h) = 2$**.** There is $1 \leq i < m$ s.t. $\mathsf{NUM}(\$_{h+1} w_i \$_{h+1}) = \mathsf{NUM}(\$_{h+1} w_m \$_{h+1})$.

An *infinite* (h,n)*-good word* w is an infinite word of the form $w \cdot \{\#\}^\omega$ such that w is a finite (h,n)-good word. The proofs of the following Lemma 4 (which is based on results of Proposition 2) and Lemma 5 can be found in [BL08].

Lemma 4. *For each $h \geq 1$ and $n \geq 1$, there is a $\mathsf{HL}_2^h(\downarrow)$ formula $\psi_{h,n}^{GOOD}$ of size $O(n^3)$ such that $\mathcal{L}(\psi_{h,n}^{GOOD})$ is the set of infinite (h,n)-good words.*

Lemma 5. *For each $n \geq 1$ and $h \geq 1$, any Büchi NWA accepting the set of infinite (h,n)-good words needs at least* $\mathsf{Tower}(n, h+1)$ *states.*

For all $n, h \geq 1$ and $k < h$, let $\psi_{h,n}^{GOOD}$ be the $\mathsf{HL}_2^h(\downarrow)$ formula of Lemma 4, and let φ be an equivalent HL^k formula. By Corollary 2, φ can be translated into an equivalent Büchi NWA $\mathcal{A}_\varphi$ of size $\mathsf{Tower}(O(|\varphi|), k+1)$. By Lemma 5, $\mathcal{A}_\varphi$ has at least $\mathsf{Tower}(n, h+1)$ states, hence $|\varphi|$ is at least $\mathsf{Tower}(\Omega(n), h-k)$. Thus, we obtain the following result.

Theorem 7. *For all $h > k \geq 0$ and $m \geq 2$, there is a succinctness gap between $\mathsf{HL}_m^h(\downarrow)$ and HL^k of exponential height $h-k$.*

5 Conclusions

There are two interesting questions which have been left open in this paper. In the HL_2^h formulas used in the proof of h-EXPSPACE-hardness, there is a strict nested alternation between the binder modalities $\exists x_1$ and $\exists x_2$. Since satisfiability of HL_1 is only EXPSPACE-complete, we conjecture that the considered decision problems can be solved in space of exponential height equal to the depth of nested alternations of binder modalities associated with distinct variables. Another interesting question is to investigate the succinctness gap between HL^k and $\mathsf{HL}^k(\downarrow)$ for each $k \geq 1$.

References

[ABM01] Areces, C., Blackburn, P., Marx, M.: Hybrid logics: Characterization, interpolation and complexity. J. Symb. Log. 66(3), 977–1010 (2001)

[BL08] Bozzelli, L., Lanotte, R.: Complexity and Succinctness issues for linear-time hybrid logics. Technical report (2008), http://dscpi.uninsubria.it/staff/Bozzelli

[BS98] Blackburn, P., Seligman, J.: What are hybrid languages? In: Kracht, de Rijke, Wansing, Zakharyaschev (eds.) Advances in Modal Logic, vol. 1, pp. 41–62. CSLI Publications (1998)

[Buc62] Buchi, J.R.: On a decision method in restricted second order arithmetic. In: Proc. Internat. Congr. Logic, Method. and Philos. Sci., Stanford, pp. 1–12 (1960)

[CF05] ten Cate, B., Franceschet, M.: On the complexity of hybrid logics with binders. In: Ong, L. (ed.) CSL 2005. LNCS, vol. 3634, pp. 339–354. Springer, Heidelberg (2005)

[DS02] Demri, S., Schnoebelen, Ph.: The complexity of propositional linear temporal logics in simple cases. Information and Computation 174(1), 84–103 (2002)

[FR06] Franceschet, M., de Rijke, M.: Model checking hybrid logics (with an application to semistructured data). J. Applied Logic 4(3), 279–304 (2006)

[FRS03] Franceschet, M., de Rijke, M., Schlingloff, B.H.: Hybrid logics on linear structures: Expressivity and complexity. In: Proc. 10th TIME, pp. 166–173. IEEE Computer Society Press, Los Alamitos (2003)

[Gor96] Goranko, V.: Hierarchies of modal and temporal logics with reference pointers. Journal of Logic, Language and Information 5(1), 1–24 (1996)

[MH84] Miyano, S., Hayashi, T.: Alternating finite automata on ω-words. Theoretical Computer Science 32, 321–330 (1984)

[Pnu77] Pnueli, A.: The temporal logic of programs. In: Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, pp. 46–57 (1977)

[Saf88] Safra, S.: On the complexity of omega-automata. In: Proc. 29th FOCS, pp. 319–327 (1988)

[SC85] Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. Journal of the ACM 32(3), 733–749 (1985)

[Sto74] Stockmeyer, L.J.: The complexity of decision problems in automata theory and logic. Ph.D. thesis, Department of Electrical Engineering. MIT, Cambridge (1974)

[SW07] Schwentick, T., Weber, V.: Bounded-variable fragments of hybrid logics. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 561–572. Springer, Heidelberg (2007)

[Var88] Vardi, M.Y.: A temporal fixpoint calculus. In: Proc. 15th Annual POPL, pp. 250–259. ACM Press, New York (1988)

Optimal Tableaux for Right Propositional Neighborhood Logic over Linear Orders*

Davide Bresolin[1], Angelo Montanari[2], Pietro Sala[2], and Guido Sciavicco[3]

[1] Department of Computer Science,
University of Verona, Verona, Italy
davide.bresolin@univr.it
[2] Department of Mathematics and Computer Science,
University of Udine, Udine, Italy
{angelo.montanari,pietro.sala}@dimi.uniud.it
[3] Department of Information, Engineering and Communications,
University of Murcia, Murcia, Spain
guido@um.es

Abstract. The study of interval temporal logics on linear orders is a meaningful research area in computer science and artificial intelligence. Unfortunately, even when restricted to propositional languages, most interval logics turn out to be undecidable. Decidability has been usually recovered by imposing severe syntactic and/or semantic restrictions. In the last years, tableau-based decision procedures have been obtained for logics of the temporal neighborhood and logics of the subinterval relation over *specific* classes of temporal structures. In this paper, we develop an optimal NEXPTIME tableau-based decision procedure for the future fragment of Propositional Neighborhood Logic over the *whole* class of linearly ordered domains.

1 Introduction

Propositional interval temporal logics play a significant role in computer science and artificial intelligence, as they provide a natural framework for representing and reasoning about temporal properties. Unfortunately, the computational complexity of most of them constitutes a barrier to their extensive use in practical applications (the two prominent interval temporal logics, namely, Halpern and Shoham's HS [8] and Venema's CDT [11], are highly undecidable). Not surprisingly, recent research in the area focused on the development of implementable deduction systems for them. Early work in this direction includes Bowman and Thompson's decision procedure for propositional ITL [9], interpreted over finite linearly ordered domains [1], and a non-terminating tableau system for CDT, interpreted over partially ordered domains [7]. In the former case, decidability is achieved by introducing a simplifying hypothesis, called *locality* principle, that

* This work has been partially supported by the European projects FP7-ICT-217069 COCONUT and FP7-ICT-223844 CON4COORD.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 62–75, 2008.

constrains the relation between the truth value of a formula over an interval and its truth values over initial subintervals of that interval.

Tableau-based decision procedures have been recently obtained for some interval temporal logics over *specific* classes of temporal structures, without resorting to any simplifying assumption.

The logic D of the subinterval relation is a fragment of HS which features a single unary modality corresponding to the strict subinterval relation, where a subinterval has no endpoints in common with the current one. In [2], Bresolin et al. devise a sound and complete PSPACE-complete tableau system for D interpreted in the class of all dense linearly ordered sets. Moreover, they extended such a result to the logic $\mathrm{D}_{\sqsubset}$, where a subinterval may have (at most) one endpoint in common with the current one. The decision problem for D over other classes of temporal structures, including the whole class of linearly ordered domains, the class of discrete linearly ordered domains, $\mathbb{N}$, and $\mathbb{Z}$, is still open.

The logic PNL of temporal neighborhood is the propositional fragment of Neighborhood Logic [6]. It can be viewed as a fragment of HS that features two modal operators $\langle A \rangle$ and $\langle \overline{A} \rangle$, that respectively correspond to the *met-by* and the *meets* relations. The logical properties of PNL have been systematically investigated in [3]. In particular, NEXPTIME-completeness of PNL when interpreted over various classes of temporal structures, including all linearly ordered domains, all well-ordered domains, all finite linearly ordered domains, and $\mathbb{N}$ [3], has been shown via a reduction to the satisfiability problem for the two-variable fragment of first-order logic for binary relational structures over ordered domains [10]. Despite these significant achievements, the problem of devising decision procedures for PNL of practical interest has been only partially solved. In [5], a tableau system for its future fragment RPNL, interpreted over $\mathbb{N}$, has been developed; such a result has been later extended to full PNL over $\mathbb{Z}$ [4].

In this paper, we focus our attention on RPNL interpreted in the whole class of linearly ordered domains, and we develop a NEXPTIME tableau system for it. Since NEXPTIME-completeness holds for PNL and its single-modality fragments [3], the proposed solution turns out to be optimal. From a technical point of view, the proposed tableau system is quite different from the one for $\mathbb{N}$ [5]. While models for RPNL formulas over $\mathbb{N}$ can be generated by simply adding future points (possibly infinitely many) to a given partial model, the construction of a model for an RPNL formula over an arbitrary linearly ordered domain may require the addition of points (possibly infinitely many) in between existing ones. Such a difference is illustrated in Section 2 by a simple example.

The paper is organized as follows. In Section 2, we introduce syntax and semantics of RPNL and we analyse its expressiveness. In Section 3, we describe a terminating tableau system for RPNL interpreted in the class of all linearly ordered domains. An example of the procedure at work concludes the section. In Section 4, we prove its soundness, completeness, and optimality. Conclusions provide an assessment of the work and outline future research directions.

2 RPNL over Linearly Ordered Domains

In this section, we first provide syntax and semantics of *Right Propositional Neighborhood Logic* (RPNL, for short); then, we show that RPNL is expressive enough to distinguish between satisfiability over $\mathbb{N}$ and over the class of all linearly ordered domains. The language of RPNL consists of a set $\mathcal{AP}$ of atomic propositions, the propositional connectives $\neg, \vee$, and the modal operator $\langle A \rangle$ (you can read $\langle A \rangle$ as adjacent). The other propositional connectives, as well as the logical constants $\top$ (*true*) and $\bot$ ($false$) and the dual modal operator $[A]$, are defined as usual. The *formulas* of RPNL, denoted by $\varphi, \psi, \ldots$, are generated by the following abstract syntax:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle A \rangle \varphi.$$

Given a linearly ordered domain $\mathbb{D} = \langle D, < \rangle$, an *interval* over $\mathbb{D}$ is an ordered pair $[d_i, d_j]$ such that $d_i < d_j$. An *interval structure* is a pair $\langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle$, where $\mathbb{I}(\mathbb{D})$ is the set of all intervals over $\mathbb{D}$. Logics of temporal neighborhood have been studied in different flavors, either including or excluding point intervals, that is, intervals of the form $[d_i, d_i]$, and a modal constant to capture them. In this paper, we assume the so-called *strict* semantics (point intervals are not admitted); however, similar results can be obtained for the non-strict case.

The semantics of RPNL is given in terms of *interpretations* of the form $\mathbf{M} = \langle \langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle, \mathcal{V} \rangle$, where $\langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle$ is an interval structure and $\mathcal{V} : \mathcal{AP} \to 2^{\mathbb{I}(\mathbb{D})}$ is a valuation function assigning a set of intervals to every atomic proposition. It is recursively defined by the satisfiability relation $\models$ as follows:

- for every $p \in \mathcal{AP}$, $\mathbf{M}, [d_i, d_j] \models p$ iff $[d_i, d_j] \in \mathcal{V}(p)$;
- $\mathbf{M}, [d_i, d_j] \models \neg\psi$ iff $\mathbf{M}, [d_i, d_j] \not\models \psi$;
- $\mathbf{M}, [d_i, d_j] \models \psi_1 \vee \psi_2$ iff $\mathbf{M}, [d_i, d_j] \models \psi_1$ or $\mathbf{M}, [d_i, d_j] \models \psi_2$;
- $\mathbf{M}, [d_i, d_j] \models \langle A \rangle \psi$ iff $\exists [d_j, d_k] \in \mathbb{I}(\mathbb{D})$ such that $\mathbf{M}, [d_j, d_k] \models \psi$.

We denote by $[A]\psi$ the formula $\neg\langle A \rangle \neg\psi$. Note that $[A]\psi$ means that every adjacent future interval must make p true, while $[A][A]\psi$ means that ψ is true over every non-adjacent future interval. Given an RPNL-formula φ, we denote by $(A)\varphi$ a formula of the form $\langle A \rangle \varphi$ or $[A]\varphi$. We define the *closure* of φ (denoted by $\mathrm{CL}(\varphi)$) as the set of all subformulas of φ (including φ itself) and of their negations, and the *temporal closure* of φ (denoted by $\mathrm{TF}(\varphi)$) as the set $\{(A)\psi \mid (A)\psi \in CL(\varphi)\}$.

To show that RPNL is expressive enough to distinguish between satisfiability over $\mathbb{N}$ and over the class of all linearly ordered domains, we exhibit a formula that is unsatisfiable over the former and satisfiable over the latter.

Let $[G]$ be the *universally-in-the-future* operator defined as follows: $[G]\psi = \psi \wedge [A]\psi \wedge [A][A]\psi$ and let seq_p be a shorthand for $p \to \langle A \rangle p$. Consider the formula $AccPoints = \langle A \rangle p \wedge [G] seq_p \wedge \langle A \rangle [G] \neg p$. We will show that $AccPoints$ is unsatisfiable over $\mathbb{N}$, while it is satisfiable whenever the temporal structure in which it is interpreted has at least one *accumulation point*, that is, a point which is the right bound of an infinite (ascending) chain of points.

Proposition 1. *The* RPNL*-formula AccPoints is satisfiable over the class of all linearly ordered domains, while it is not satisfiable over* $\mathbb{N}$.

Proof. We first show that the formula *AccPoints* is not satisfiable over $\mathbb{N}$. Suppose, by contradiction, that there exists an interpretation $\mathbf{M}$, based on $\mathbb{N}$, such that $\mathbf{M}, [d_0, d_1] \models \mathit{AccPoints}$. From $\mathbf{M}, [d_0, d_1] \models \langle A\rangle p \wedge [G]\mathit{seq}_p$, it follows that there exists a sequence of points $d_1 < d_{j_1} < d_{j_2} \ldots$ such that $\mathbf{M}, [d_1, d_{j_1}] \models p$ and $\mathbf{M}, [d_{j_i}, d_{j_{i+1}}] \models p$, for all $i \geq 1$. Moreover, from $\mathbf{M}, [d_0, d_1] \models \langle A\rangle[G]\neg p$, it follows that there exists a point d_i such that $\mathbf{M}, [d_1, d_i] \models [G]\neg p$. Two cases may arise.

Case (1). Suppose $d_i < d_{j_1}$. From $\mathbf{M}, [d_1, d_i] \models [A][A]\neg p$, it follows that $\mathbf{M}, [d_i, d_{j_1}] \models [A]\neg p$ and thus $\mathbf{M}, [d_{j_1}, d_{j_2}] \models \neg p$. This allows us to conclude that both p and $\neg p$ hold over $[d_{j_1}, d_{j_2}]$ as shown in Figure 1.

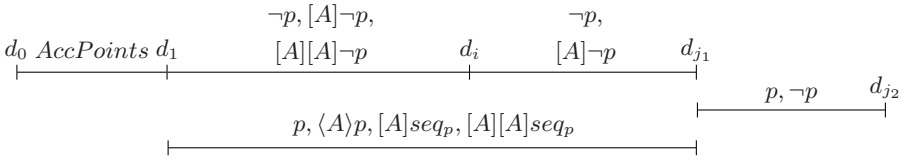


Fig. 1. Unsatisfiability of *AccPoints* over $\mathbb{N}$: case (1)

Case (2). Suppose $d_{j_1} < d_i$. From $\mathbf{M}, [d_1, d_i] \models [A][A]\neg p$, it follows that, for any point $d_k > d_i$, $\mathbf{M}, [d_i, d_k] \models [A]\neg p$ and, for any point $d_m > d_k$, $\mathbf{M}, [d_k, d_m] \models \neg p$. Since *AccPoints* is interpreted over $\mathbb{N}$, there exists a point $d_{j_h} > d_i$ such that p holds over $[d_{j_h}, d_{j_{h+1}}]$. Hence, both p and $\neg p$ hold over $[d_{j_h}, d_{j_{h+1}}]$ as shown in Figure 2.

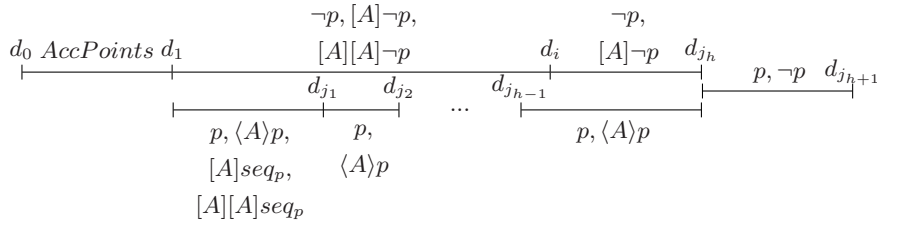


Fig. 2. Unsatisfiability of *AccPoints* over $\mathbb{N}$: case (2)

Let us consider now the class of all linearly ordered domains. A model satisfying *AccPoints* can be built as follows: we take an infinite sequence of points $d_{j_1} < d_{j_2} < d_{j_3} < \ldots$ such that $\mathbf{M}, [d_{j_i}, d_{j_{i+1}}] \models p$, for every $i \geq 1$, and then we add an accumulation point d_ω greater than d_{j_i}, for every $i \geq 1$, such that $\mathbf{M}, [d_1, d_\omega] \models [G]\neg p$. The definition of the valuation function can be easily completed without introducing any contradiction, thus showing that *AccPoints* is satisfiable (see Figure 3). □

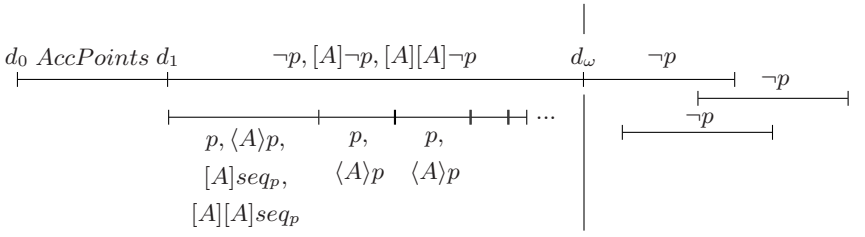


Fig. 3. A model for *AccPoints* over the class of linearly ordered domains

As shown by Proposition 1, RPNL interpreted over $\mathbb{N}$ differs from RPNL interpreted in the class of all linearly ordered domains. This prevents us from exploiting the tableau-based decision procedure for RPNL over $\mathbb{N}$ developed in [5] to check the satisfiability of RPNL formulas over the class of all linearly ordered domains. If applied to the formula *AccPoints*, such a procedure would correctly answer 'unsatisfiable', as there are no models satisfying it based on $\mathbb{N}$. In the next section, we devise an original tableau system with the ability of dealing with accumulation points.

3 A Tableau for RPNL over Linearly Ordered Domains

In the following, we first define the structure of a tableau for an RPNL-formula and then we show how to construct it. A tableau for an RPNL formula is a suitable *labeled tree* $\mathcal{T}$. Every node n of $\mathcal{T}$ is labeled by a tuple $\nu(n) = \langle [d_i, d_j], \Gamma_n, \mathbb{D}_n \rangle$, where $\mathbb{D}_n$ is a *finite* linear order, $[d_i, d_j] \in \mathbb{I}(\mathbb{D}_n)$, and $\Gamma_n \subseteq \mathrm{CL}(\varphi)$.

Expansion rule. The expansion rule adds new nodes at the end of the branch to which it is applied. Given a branch B, $B \cdot n_1$ denotes the result of expanding B with the node n_1, while $B \cdot n_1 | \ldots | n_k$ denotes the result of adding k immediate successors nodes $n_1, \ldots, n_k$ to B. A node n in a branch B such that the interval component $[d_i, d_j]$ of its labeling does not belong to the labeling of any other node in B is called an *active* node. In general, the same interval $[d_i, d_j]$ may belong to (the labeling of) different nodes in a branch B. In such a case, the farthest-from-the-root node in B labeled with $[d_i, d_j]$ is the *active* one, while the others are *non-active*. The expansion rule can be applied to active nodes only. With a little abuse of notation, given a branch B, we write $([d_i, d_j], \psi) \in B$ if there exists a node n in B, labeled with $\langle [d_i, d_j], \Gamma_n, \mathbb{D}_n \rangle$, such that $\psi \in \Gamma_n$.

Definition 1. *Let B be a branch, $\mathbb{D}_B$ be the linearly ordered set belonging to the label of the leaf of B, and n be an active node in B with label $\nu(n) = \langle [d_i, d_j], \Gamma_n, \mathbb{D}_n \rangle$. The* expansion rule *for $n \in B$ is defined case-by-case as follows:*

OR: if $\psi_1 \vee \psi_2 \in \Gamma_n$, $\psi_1 \notin \Gamma_n$, and $\psi_2 \notin \Gamma_n$, expand B to $B \cdot n_1 | n_2$, with $\nu(n_1) = \langle [d_i, d_j], \Gamma_n \cup \{\psi_1\}, \mathbb{D}_B \rangle$ and $\nu(n_2) = \langle [d_i, d_j], \Gamma_n \cup \{\psi_2\}, \mathbb{D}_B \rangle$;

AND: if $\neg(\psi_1 \vee \psi_2) \in \Gamma_n$ *and* $\neg\psi_1 \notin \Gamma_n$ *or* $\neg\psi_2 \notin \Gamma_n$*, expand* B *to* $B \cdot n_1$*, with* $\nu(n_1) = \langle [d_i, d_j], \Gamma_n \cup \{\neg\psi_1, \neg\psi_2\}, \mathbb{D}_B \rangle$*;*

NOT: if $\neg\neg\psi \in \Gamma_n$ *and* $\psi \notin \Gamma_n$*, expand* B *to* $B \cdot n_1$*, with* $\nu(n_1) = \langle [d_i, d_j], \Gamma_n \cup \{\psi\}, \mathbb{D}_B \rangle$*;*

DIAMOND: if $\langle A \rangle \psi \in \Gamma_n$ *and there exists no* $d_k \in \mathbb{D}_B$ *such that* $d_k > d_j$ *and* $([d_j, d_k], \psi) \in B$*, then proceed as follows. Let* $d_{j+1}, \ldots, d_{j+k}$ *be the points in* $\mathbb{D}_B$ *greater than* d_j*. Expand* B *to* $B \cdot n_1 | \ldots | n_k | m_0 | \ldots | m_{k+1}$*, where*

1. *for every* $1 \leq h \leq k$*, if there exists an active node* $n' \in B$ *labeled with* $\langle [d_j, d_{j+h}], \Gamma', \mathbb{D}' \rangle$*, then* $\nu(n_h) = \langle [d_j, d_{j+h}], \Gamma' \cup \{\psi\}, \mathbb{D}_B \rangle$*, otherwise* $\nu(n_h) = \langle [d_j, d_{j+h}], \{\psi\}, \mathbb{D}_B \rangle$*;*
2. *for every* $0 \leq h \leq k$*,* $\nu(m_h) = \langle [d_j, d^h], \{\psi\}, \mathbb{D}_B^h \rangle$*, where* $\mathbb{D}_B^h$ *is obtained from* $\mathbb{D}_B$ *by adding a new point* d^h *strictly in between* d_{j+h} *and* d_{j+h+1} *(if* $h = k$*, the second condition is obviously missing);*

BOX: if $\neg\langle A \rangle \psi \in \Gamma_n$ *and there exists* $d_k \in \mathbb{D}_B$ *such that* $d_j < d_k$ *and* $([d_j, d_k], \neg\psi) \notin B$*, expand* B *to* $B \cdot n_1$*. If there are no nodes in* B *labeled with* $[d_j, d_k]$*, then* $\nu(n_1) = \langle [d_j, d_k], \{\neg\psi\}, \mathbb{D}_B \rangle$*; otherwise,* $\nu(n_1) = \langle [d_j, d_k], \Gamma_{n'} \cup \{\neg\psi\}, \mathbb{D}_B \rangle$*, where* n' *is the (unique) active node in* B *labeled with* $[d_j, d_k]$*.*

Blocking condition. Given a branch B and a point $d' \in \mathbb{D}_B$, the set $\mathrm{REQ}(d')$ of the *temporal requests* of d' is defined as follows:

$$\mathrm{REQ}(d') = \{\psi \in \mathrm{TF}(\varphi) : \exists [d'', d'], ([d'', d'], \psi) \in B\}.$$

Moreover, we define the set of *past temporal requests* of d as the following set of sets:

$$\mathrm{PAST}(d) = \{\mathrm{REQ}(d') : d' < d\}.$$

Definition 2. *We say that a point* $d_i \in \mathbb{D}_B$ *is* blocked *if there exists a point* $d_j \in \mathbb{D}_B$*, with* $d_j < d_i$*, such that (i)* $\mathrm{REQ}(d_i) = \mathrm{REQ}(d_j)$ *and (ii)* $\mathrm{PAST}(d_i) = \mathrm{PAST}(d_j)$*.*

Notice that for any pair d_i, d_j, $\mathrm{PAST}(d_i) = \mathrm{PAST}(d_j)$ if and only if for every $d' < d_i$ there exists $d'' < d_j$ such that $\mathrm{REQ}(d') = \mathrm{REQ}(d'')$.

Expansion strategy. We say that a branch B is *closed* if there exist a formula ψ and an interval $[d_i, d_j]$ such that both $([d_i, d_j], \psi) \in B$ and $([d_i, d_j], \neg\psi) \in B$, otherwise we say that B is *open*. Moreover, we say that an expansion rule is *applicable* to a node n if n is active and its application generates at least one node with a new labeling.

Definition 3. *The* expansion strategy *for a branch* B *is defined as follows:*

i) Apply the expansion rule to B *only if* B *is open;*

ii) If B *is open, apply the AND, OR, NOT, and BOX rules to the closest-to-the-root active node to which the expansion rule is applicable;*

iii) If B *is open, apply the DIAMOND rule to the closest-to-the-root active node to which the expansion rule is applicable, provided that the right endpoint of the interval* $[d_i, d_j]$ *in its labeling is not blocked.*

Definition 4. *A* tableau *for an* RPNL*-formula φ is any finite labeled tree obtained by expanding the one-node labeled tree $\langle [d_0, d_1], \{\varphi\}, \{d_0 < d_1\}\rangle$ through successive application of the branch-expansion strategy to existing branches, until it cannot be applied anymore.*

Open and closed tableau. Given a formula φ and a tableau $\mathcal{T}$ for it, we say that $\mathcal{T}$ is *closed* if all its branches are closed, otherwise it is *open*.

Example. We illustrate the behaviour of the proposed tableau system by applying it to the formula $\varphi = \langle A\rangle p \wedge \langle A\rangle[A]\neg p \wedge [A](p \rightarrow \langle A\rangle p)$. A portion of the resulting tableau is depicted in Figure 4. For the sake of readability, we will describe sequences of expansion steps that do not split the branch (applications of the AND, NOT, and BOX rules) as single expansion steps. Moreover, instead of explicitly representing the linear orders associated with the nodes, we will simply display the extensions to the linear order when they are introduced. Finally, in the textual explanation we will identify a branch with its leaf node.

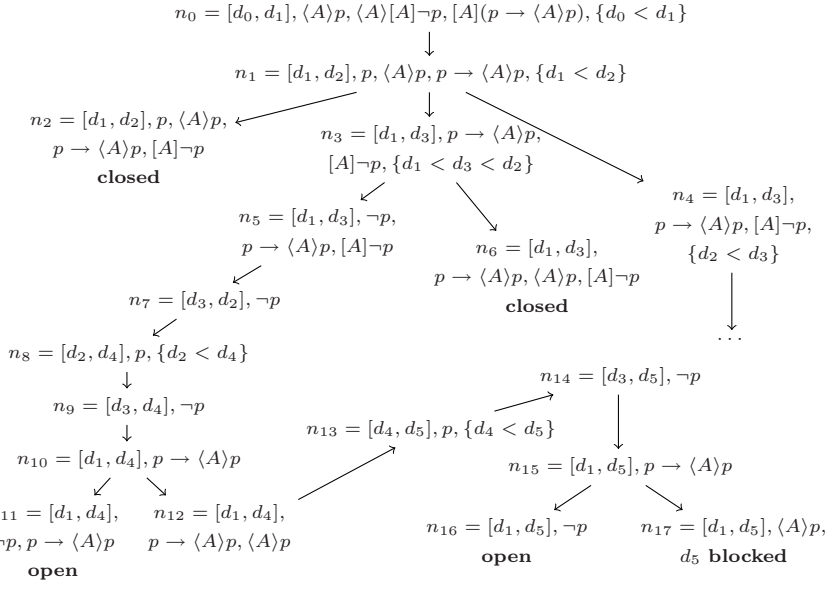


Fig. 4. Part of the tableau for the formula $\varphi = \langle A\rangle p \wedge \langle A\rangle[A]\neg p \wedge [A](p \rightarrow \langle A\rangle p)$

The root n_0 of the tableau contains the $\langle A\rangle$-formulas $\langle A\rangle p$ and $\langle A\rangle[A]\neg p$. We first apply the DIAMOND rule to $\langle A\rangle p$. Since d_1 is the greatest point of the current linear order, we can only add a point d_2 to the right of d_1 and satisfy p over the interval $[d_1, d_2]$ (node n_1). (In fact, node n_1 is obtained by an application of the DIAMOND rule followed by an application of the OR rule and the removal of the inconsistent node including both p and $\neg p$.) Next, we apply the DIAMOND rule to the formula $\langle A\rangle[A]\neg p$ in n_0 and generate the nodes n_2,

n_3, and n_4. The node n_2 is closed, because it contains both $\langle A\rangle p$ and $[A]\neg p$. The expansion proceeds by the application of the OR rule to n_3, that generates the nodes n_5 and n_6. The application of the BOX rule to n_5 generates the node n_7, which is further expanded by applying the DIAMOND rule to the formula $\langle A\rangle p$ in n_1. With two applications of the BOX rule (to nodes n_5 and n_0, respectively), we generate the node n_{10}. Then, the application of the OR rule to n_{10} generates the nodes n_{11} and n_{12}. The branch ending in n_{11} can be easily shown to be open, because all the $\langle A\rangle$ formulas in it are fulfilled and no more expansion rules are applicable to it. Such a condition allows us to conclude that the formula φ is satisfiable. To give an example of the application of the blocking condition, we expand the tableau a bit more. By applying the DIAMOND rule to n_{12}, we obtain the node n_{13}. Two applications of the BOX rule to n_{13} generates the nodes n_{14} and n_{15}. The OR rule is then applied to n_{15}. The branch ending in n_{16} is open, because all $\langle A\rangle$ formulas are fulfilled and no expansion rules can be applied to it. The branch ending in n_{17} is not expanded anymore, because point d_5 is blocked ($REQ(d_5) = REQ(d_4)$ and $PAST(d_5) = PAST(d_4)$).

4 Soundness, Completeness, and Complexity

In this section, we show that the proposed tableau method is sound, complete, and terminating. In addition, we prove that it is complexity optimal. As a preliminary step, we recall some basic notions (details can be found in [5]).

Definition 5. *A* φ-atom *is a set* $A \subseteq \mathrm{CL}(\varphi)$ *such that, for every* $\psi \in \mathrm{CL}(\varphi)$, $\psi \in A$ *iff* $\neg\psi \notin A$, *and, for every* $\psi_1 \vee \psi_2 \in \mathrm{CL}(\varphi)$, $\psi_1 \vee \psi_2 \in A$ *iff* $\psi_1 \in A$ *or* $\psi_2 \in A$.

We denote the set of all φ-atoms by A_φ. Atoms are connected by the following binary relation.

Definition 6. *Let* R_φ *be a binary relation over* A_φ *such that, for every pair of atoms* $A, A' \in A_\varphi$, $A\ R_\varphi\ A'$ *if and only if, for every* $[A]\psi \in \mathrm{CL}(\varphi)$, *if* $[A]\psi \in A$, *then* $\psi \in A'$.

We now introduce a suitable labeling of interval structures based on φ-atoms.

Definition 7. *A* φ-labeled interval structure *(*LIS *for short) is a pair* $\mathbf{L} = \langle\langle \mathbb{D}, \mathbb{I}(\mathbb{D})\rangle, \mathcal{L}\rangle$, *where* $\langle \mathbb{D}, \mathbb{I}(\mathbb{D})\rangle$ *is an interval structure and* $\mathcal{L} : \mathbb{I}(\mathbb{D}) \to A_\varphi$ *is a* labeling function *such that, for every pair of neighboring intervals* $[d_i, d_j], [d_j, d_k]$, $\mathcal{L}([d_i, d_j])\ R_\varphi\ \mathcal{L}([d_j, d_k])$.

If we interpret the labeling function as a valuation function, LISs represent *candidate models* for φ. The truth of formulas devoid of temporal operators and that of $[A]$-formulas indeed follow from the definition of φ-atom and the definition of R_φ, respectively. However, to obtain a model for φ we must also guarantee the truth of $\langle A\rangle$-formulas.

Definition 8. *A φ-labeled interval structure* $\mathbf{L} = \langle\langle \mathbb{D}, \mathbb{I}(\mathbb{D})\rangle, \mathcal{L}\rangle$ *is* fulfilling *if and only if, for every temporal formula* $\langle A\rangle\psi \in \mathrm{TF}(\varphi)$ *and every interval* $[d_i, d_j] \in \mathbb{I}(\mathbb{D})$*, if* $\langle A\rangle\psi \in \mathcal{L}([d_i, d_j])$*, then there exists* $d_k > d_j$ *such that* $\psi \in \mathcal{L}([d_j, d_k])$.

Theorem 1 (Fulfilling LISs and satisfiability [5]). *A formula φ is satisfiable if and only if there exists a fulfilling* LIS $\mathbf{L} = \langle\langle \mathbb{D}, \mathbb{I}(\mathbb{D})\rangle, \mathcal{L}\rangle$ *with* $\varphi \in \mathcal{L}([d_0, d_1])$.

Theorem 2 (Soundness). *If φ is satisfiable, then the tableau for it is open.*

Proof. Let φ be a satisfiable formula and let $\mathbf{L} = \langle\langle \mathbb{D}, \mathbb{I}(\mathbb{D})\rangle, \mathcal{L}\rangle$ be a fulfilling LIS for φ. We show how an open tableau $\mathcal{T}$ for φ can be obtained from $\mathbf{L}$. Since $\mathbf{L}$ is a fulfilling LIS for φ, there exists an interval $[d_0, d_1]$ such that $\varphi \in \mathcal{L}([d_0, d_1])$. We start the construction of $\mathcal{T}$ with the one-node initial tableau $\langle [d_0, d_1], \{\varphi\}, \{d_0 < d_1\}\rangle$ and then we proceed in accordance with the expansion strategy. We prove by induction on the number of steps of the tableau construction that the current tableau $\mathcal{T}$ includes a branch B which satisfies the following invariant:

$$\text{for every } n = \langle [d_i, d_j], \Gamma_n, \mathbb{D}_n\rangle \text{ in } B \text{ and every } \psi \in \Gamma_n,\ \psi \in \mathcal{L}([d_i, d_j]).$$

By construction, the initial tableau satisfies the invariant. As for the inductive step, let $\mathcal{T}$ be the current tableau and let B be the branch of $\mathcal{T}$ that satisfies the invariant. Moreover, let $n = \langle [d_i, d_j], \Gamma_n, \mathbb{D}_n\rangle$ be the node in B taken into consideration by the expansion strategy. The following cases may arise:

- The OR rule is applied to n. We have that $\psi_1 \vee \psi_2 \in \Gamma_n$, $\psi_1 \notin \Gamma_n$, and $\psi_2 \notin \Gamma_n$. By the inductive hypothesis, $\psi_1 \vee \psi_2 \in \mathcal{L}([d_i, d_j])$. By definition of $\mathcal{L}$, there exists ψ_k, with $k \in \{1, 2\}$, such that $\psi_k \in \mathcal{L}([d_i, d_j])$. The expansion of B into $B \cdot n_k$, with $\nu(n_k) = \langle [d_i, d_j], \Gamma_n \cup \{\psi_k\}, \mathbb{D}_B\rangle$, maintains the invariant true.
- The AND rule is applied to n. We have that $\neg(\psi_1 \vee \psi_2) \in \Gamma_n$ and $\neg\psi_1 \notin \Gamma_n$ or $\neg\psi_2 \notin \Gamma_n$. By the inductive hypothesis, $\neg(\psi_1 \vee \psi_2) \in \mathcal{L}([d_i, d_j])$. By definition of $\mathcal{L}$, both $\neg\psi_1 \in \mathcal{L}([d_i, d_j])$ and $\neg\psi_2 \in \mathcal{L}([d_i, d_j])$. It immediately follows that the expanded branch $B \cdot n_1$, with $\nu(n_1) = \langle [d_i, d_j], \Gamma_n \cup \{\neg\psi_1, \neg\psi_2\}, \mathbb{D}_B\rangle$, preserves the invariant.
- The NOT rule is applied to n. We have that $\neg\neg\psi \in \Gamma_n$ and $\psi \notin \Gamma_n$. By the inductive hypothesis, $\neg\neg\psi \in \mathcal{L}([d_i, d_j])$ and thus $\psi \in \mathcal{L}([d_i, d_j])$. The expanded branch $B \cdot n_1$, with $\nu(n_1) = \langle [d_i, d_j], \Gamma_n \cup \{\psi\}, \mathbb{D}_B\rangle$, satisfies the invariant.
- The DIAMOND rule is applied to n. We have that $\langle A\rangle\psi \in \Gamma_n$ and there exists no $d_k \in \mathbb{D}_B$ such that $d_k > d_j$ and $([d_j, d_k], \psi) \in B$. Since $\langle A\rangle\psi \in \mathcal{L}([d_i, d_j])$, by definition of fulfilling LIS, $\psi \in \mathcal{L}([d_j, d'])$ for some $d' > d_j$ in $\mathbb{D}$. Two cases may arise:
  - $d' \in \mathbb{D}_B$. If there are no nodes in B labeled with $[d_j, d']$, we expand B with a node $\overline{n}$, with $\nu(\overline{n}) = \langle [d_j, d'], \{\psi\}, \mathbb{D}_B\rangle$; otherwise, we expand B with a node $\overline{n}$, with $\nu(\overline{n}) = \langle [d_j, d'], \Gamma_{n'} \cup \{\psi\}, \mathbb{D}_B\rangle$ where n' is the (unique) active node in B labeled with $[d_j, d']$. In both cases, the expansion of B with $\overline{n}$ preserves the invariant.

- $d' \notin \mathbb{D}_B$. Let $d_{j+1}, \ldots, d_{j+h}$, with $h \geq 0$, be the points in $\mathbb{D}_B$ greater than d_j. We have that $d_{j+k} < d' < d_{j+k+1}$ for some $0 \leq k \leq h$. Let $\mathbb{D}'$ be the linear order obtained from $\mathbb{D}_B$ by putting the new point d' in between d_{j+k} and d_{j+k+1}. The expansion of B with a node m_k, with $\nu(m_k) = \langle [d_j, d'], \{\psi\}, \mathbb{D}' \rangle$, preserves the property.

– The BOX rule is applied to n. We have that $\neg\langle A \rangle \psi \in \Gamma_n$ and there exists $d_k \in \mathbb{D}_B$ such that $d_j < d_k$, and $([d_j, d_k], \neg\psi) \notin B$. Since $\neg\langle A \rangle \psi = [A]\neg\psi \in \mathcal{L}([d_i, d_j])$, $\neg\psi \in \mathcal{L}([d_j, d_h])$ for all $d_j < d_h$, and thus $\neg\psi \in \mathcal{L}([d_j, d_k])$. It can be easily seen that the expansion of B with node n_1, whose labeling is defined according to the BOX rule, preserves the truth of the invariant.

The above argument guarantees that the resulting tableau $\mathcal{T}$ features (at least) one branch B that satisfies the invariant. Suppose now, by contradiction, that B is closed. This implies that there exist an interval $[d_i, d_j]$ and a formula $\psi \in \mathrm{CL}(\varphi)$ such that both $([d_i, d_j], \psi) \in B$ and $([d_i, d_j], \neg\psi) \in B$. Given the truth of the invariant, it follows that both $\psi \in \mathcal{L}([d_i, d_j])$ and $\neg\psi \in \mathcal{L}([d_i, d_j])$ (contradiction). Hence, B is open and thus, by definition, the tableau for φ is open. □

Completeness is proved by showing how to construct a fulfilling LIS satisfying φ from a fulfilling branch B in a tableau $\mathcal{T}$ for φ. From Theorem 1, it follows immediately that φ has a model. We will take advantage of the following lemma.

Lemma 1. *Let B be an open branch of a tableau. For every $[d_i, d_j] \in \mathbb{I}(\mathbb{D}_B)$, the following conditions hold:*

- *For any $\psi \in \mathrm{CL}(\varphi)$, it never happens that both $(\psi, [d_i, d_j]) \in B$ and $(\neg\psi, [d_i, d_j]) \in B$;*
- *If $(\psi_1 \vee \psi_2, [d_i, d_j]) \in B$, then $(\psi_1, [d_i, d_j]) \in B$ or $(\psi_2, [d_i, d_j]) \in B$;*
- *If $(\neg(\psi_1 \vee \psi_2), [d_i, d_j]) \in B$, then $(\neg\psi_1, [d_i, d_j]) \in B$ and $(\neg\psi_2, [d_i, d_j]) \in B$;*
- *If $(\neg\neg\psi, [d_i, d_j]) \in B$, then $(\psi, [d_i, d_j]) \in B$;*
- *If $(\langle A \rangle \psi, [d_i, d_j]) \in B$ and d_j is not a blocked point, then there exists $d_k > d_j$ such that $(\psi, [d_j, d_k]) \in B$;*
- *If $(\neg\langle A \rangle \psi, [d_i, d_j]) \in B$, then, for every $d_k > d_j$, $(\neg\psi, [d_j, d_k]) \in B$.*

Proof. The thesis follows from the tableau rules and the expansion strategy. □

Theorem 3 (Completeness). *If the tableau for φ is open, then φ is satisfiable.*

Proof. Let $\mathcal{T}$ be the tableau for φ. By definition, if $\mathcal{T}$ is open, then there exists an open branch B in $\mathcal{T}$. We distinguish two cases.

1. There are no blocked points in B. A fulfilling LIS $\mathbf{L} = \langle \langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle, \mathcal{L} \rangle$ can be obtained as follows. As a first step, we execute the following operations:
   - $\mathbb{D} = \mathbb{D}_B$;
   - for every $\psi \in \mathrm{CL}(\varphi)$ such that $([d_0, d_1], \psi) \in B$, we let $\psi \in \mathcal{L}([d_0, d_1])$;
   - for every $[d_i, d_j] \in \mathbb{I}(\mathbb{D}_B)$, with $d_i > d_0$, and every $\psi \in \mathrm{CL}(\varphi)$ such that $([d_i, d_j], \psi) \in B$, we let $\psi \in \mathcal{L}([d_i, d_j])$;

– for every $d_j > d_1$, we let $\mathcal{L}([d_0, d_j]) = \mathcal{L}([d_1, d_j])$ (notice that for every $[d_0, d_j] \in \mathbb{I}(\mathbb{D}_B)$, with $d_j > d_1$, there are no nodes in B with an interval $[d_0, d_j]$ in their label.

The resulting structure is not necessarily a LIS: it could be the case that, for some interval $[d_i, d_j]$ and formula $\psi \in \mathrm{CL}(\varphi)$, neither $\psi \in \mathcal{L}([d_i, d_j])$ nor $\neg\psi \in \mathcal{L}([d_i, d_j])$. However, it can be extended to a complete LIS as follows (we proceed by induction on the structure of ψ):

– If $\psi = p$ or $\psi = \neg p$, we let $\neg p \in \mathcal{L}([d_i, d_j])$;
– If $\psi = \psi_1 \vee \psi_2$, we let $\psi \in \mathcal{L}([d_i, d_j])$ if and only if $\psi_1 \in \mathcal{L}([d_i, d_j])$ or $\psi_2 \in \mathcal{L}([d_i, d_j])$;
– If $\psi = \neg\psi_1$, we let $\psi \in \mathcal{L}([d_i, d_j])$ if and only if $\psi_1 \notin \mathcal{L}([d_i, d_j])$;
– If $\psi = \langle A\rangle\psi_1$, we let $\psi \in \mathcal{L}([d_i, d_j])$ if and only if there exists $d_k > d_j$ such that $\psi_1 \in \mathcal{L}([d_j, d_k])$.

Such a completion procedure produces a fulfilling LIS $\mathbf{L}$: by Lemma 1, for each pair of neighboring intervals $[d_i, d_j], [d_j, d_k]$, if $[A]\psi \in \mathcal{L}([d_i, d_j])$, then $\psi \in \mathcal{L}([d_j, d_k])$. Moreover, since there are no blocked points in B, $\mathbf{L}$ is fulfilling. By Theorem 1, we can conclude that φ is satisfiable.

2. There is at least one blocked point in B. We proceed as in the previous case. The resulting structure is in general not fulfilling. We can turn it into a fulfilling LIS $\mathbf{L}$ as follows. Let $(b_1, \ldots, b_k)$ be the list of all blocked points in $\mathbb{D}_B$, arranged in an arbitrary order. Consider now a specific point $b_i \in \{b_1, \ldots, b_k\}$ and assume that $\langle A\rangle\psi_1, \ldots, \langle A\rangle\psi_n$ are the $\langle A\rangle$-formulas in $\mathrm{REQ}(b_i)$ which are not fulfilled by the current structure. Let m_i be the *maximal blocking point* for b_i, that is, the greatest non-blocked point in $\mathbb{D}_B$ such that $m_i < b_i$, $\mathrm{REQ}(b_i) = \mathrm{REQ}(m_i)$, and $\mathrm{PAST}(b_i) = \mathrm{PAST}(m_i)$. We have that, for every $\langle A\rangle\psi_j \in \mathrm{REQ}(b_i)$ there exists an interval $[m_i, d_{\psi_j}]$ such that $\psi_j \in \mathcal{L}([m_i, d_{\psi_j}])$. Two cases may arise:

– $d_{\psi_j} > b_i$. In such a case, we fulfill $\langle A\rangle\psi_j \in \mathrm{REQ}(b_i)$ by replacing the current labeling of $[b_i, d_{\psi_j}]$ with $\mathcal{L}([b_i, d_{\psi_j}]) = \mathcal{L}([m_i, d_{\psi_j}])$;
– $m_i < d_{\psi_j} \leq b_i$. In such a case, we add a new point e_{ψ_j} in between b_i and its immediate successor (if any) and we let $\mathcal{L}([b_i, e_{\psi_j}]) = \mathcal{L}([m_i, d_{\psi_j}])$.

Such a construction must be repeated for all $\langle A\rangle\psi_j \in \{\langle A\rangle\psi_1, \ldots, \langle A\rangle\psi_n\}$. At the end, there can be a number of intervals generated by the new points, with an incomplete labeling. Let e_{ψ_j} be one of the new points. We complete the labeling of the interval starting/ending at e_{ψ_j} as follows:

– for every interval $[e_{\psi_j}, d]$, we let $\mathcal{L}([e_{\psi_j}, d]) = \mathcal{L}([d_{\psi_j}, d])$;
– for every interval $[d, e_{\psi_j}]$, with $d < d_{\psi_j}$, we let $\mathcal{L}([d, e_{\psi_j}]) = \mathcal{L}([d, d_{\psi_j}])$;
– for every interval $[d, e_{\psi_j}]$, with $d_{\psi_j} \leq d < e_{\psi_j}$, let d' be a point such that $d' < d_{\psi_j}$ and $\mathrm{REQ}(d') = \mathrm{REQ}(d)$. Since $\mathrm{PAST}(b_i) = \mathrm{PAST}(m_i)$, such point is guaranteed to exists. We let $\mathcal{L}([d, e_{\psi_j}]) = \mathcal{L}([d', d_{\psi_j}])$.

At the end of this completion process, we remove b_i from the list of blocked points. For every added point e_{ψ_j}, if there there exists a $\langle A\rangle$-formula in $\mathrm{REQ}(e_{\psi_j})$ whose request is not fulfilled by the current structure, we insert e_{ψ_j} at the end of the current list of blocked points.

By possibly repeating the above expansion step infinitely many times, we guarantee that every point added to the list of blocked point is eventually expanded. The resulting (limit) structure is thus a fulfilling labeled structure, and, by Theorem 1, we can conclude that φ is satisfiable. □

To determine the computational complexity of the proposed method, we take advantage of the following lemma.

Lemma 2. *The length of every branch in a tableau $\mathcal{T}$ for φ is bounded by $|\varphi|^3 \cdot 3^{2|\varphi|}$.*

Proof. Let B be a branch of $\mathcal{T}$ and $\mathbb{D}_B = \langle D_B, < \rangle$ be the linear order associated with it. For every $[d_i, d_j] \in \mathbb{I}(\mathbb{D}_B)$, there exists at most one active node in B labeled with $[d_i, d_j]$. Moreover, according to the expansion rule, a node n becomes inactive when a new active node n' is added to B, with $\Gamma_n \subset \Gamma_{n'}$. Hence, for every $[d_i, d_j] \in \mathbb{I}(\mathbb{D}_B)$, there exist at most $|\varphi|$ nodes in B, where $|\varphi|$ denotes the number of elements (and thus of subformulas) of φ.

The only rule that adds new points to $\mathbb{D}_B$ is the DIAMOND rule, which is applicable only to nodes labeled with intervals $[d_i, d_j]$ whose right endpoint is not blocked (according to Definition 2, a point d_j is blocked if there exists a point $d_i < d_j$ such that $\mathrm{REQ}(d_i) = \mathrm{REQ}(d_j)$ and $\mathrm{PAST}(d_i) = \mathrm{PAST}(d_j)$). For every formula $\langle A \rangle \psi \in \mathrm{TF}(\varphi)$, either $\langle A \rangle \psi \in \mathrm{REQ}(d_j)$ or $[A] \neg \psi \in \mathrm{REQ}(d_j)$ or neither $\langle A \rangle \psi$ nor $[A] \neg \psi$ belongs to $\mathrm{REQ}(d_j)$. Hence, the number of different sets of requests that can be associated with a point is less than or equal to $3^{\frac{|\mathrm{TF}|}{2}}$. Along the branch B, the PAST sets associated with points are obviously ordered by inclusion, that is, given $d, d' \in D_B$, with $d < d'$, $\mathrm{PAST}(d) \subseteq \mathrm{PAST}(d')$. This implies that the number of different PAST sets that may occur in B is bounded by the number of REQ sets (that is, $3^{\frac{|\mathrm{TF}|}{2}}$). Hence, the number of non-blocked points is less than or equal to $3^{\frac{|\mathrm{TF}|}{2}} \cdot 3^{\frac{|\mathrm{TF}|}{2}} = 3^{|\mathrm{TF}|}$.

Now, for every non-blocked point d_i, the number of $\langle A \rangle$-formulas in $\mathrm{REQ}(d_i)$ is less than or equal to $\frac{|\mathrm{TF}|}{2}$. Hence, by the application of the DIAMOND rule, every non-blocked point can introduce at most $\frac{|\mathrm{TF}|}{2}$ new points. In the worst case, the total number of points in $\mathbb{D}_B$ is thus $\frac{|\mathrm{TF}|}{2} \cdot 3^{|\mathrm{TF}|}$. This implies that the number of intervals in $\mathbb{I}(\mathbb{D}_B)$ is bounded by $\frac{|\mathrm{TF}|^2}{4} \cdot 3^{2|\mathrm{TF}|}$, and that the number of nodes in B is bounded by $|\varphi| \cdot \frac{|\mathrm{TF}|^2}{4} \cdot 3^{2|\mathrm{TF}|} \leq |\varphi|^3 \cdot 3^{2|\varphi|}$. □

Lemma 2 allows us to conclude that the satisfiability of an RPNL formula φ can be checked by a non-deterministic procedure that generates an open branch of the tableau, if any, in non-deterministic exponential time. From the NEXPTIME-hardness of the satisfiability problem for RPNL [5], the optimality of the proposed tableau method immediately follows.

Theorem 4. *The tableau method for* RPNL *interpreted over all linearly ordered domains is optimal.*

5 Conclusions

In this paper, we focused our attention on the logic RPNL. Its decidability over various classes of linear orders immediately follows from results in [3]. A limited research effort was devoted to the development of decision procedures for RPNL to be used in practice: an optimal tableau method for RPNL over $\mathbb{N}$ is given in [5] (later extended to full PNL over $\mathbb{Z}$ in [4]). In this paper, we devise a computationally optimal tableau method for RPNL interpreted in the whole class of linearly ordered domains, which turned out to be substantially different from that for $\mathbb{N}$. We are currently investigating the possibility of generalizing the proposed tableau method to cope with full PNL. Besides additional rules for the past-time modalities $\langle\overline{A}\rangle$ and $[\overline{A}]$, a revision of the definition of blocked points is needed, to distinguish between *right-blocked* (points that do not require the addition of new points to their future) and *left-blocked* (points that do not require the addition of new points to their past) points. These modifications have a relevant impact on the soundness, completeness, and termination of the method. In parallel, we are exploring the possibility of adapting the tableau method to the case of RPNL (and PNL) over dense linearly ordered domains, whose decision problem is still open.

References

1. Bowman, H., Thompson, S.: A decision procedure and complete axiomatization of finite interval temporal logic with projection. Journal of Logic and Computation 13(2), 195–239 (2003)
2. Bresolin, D., Goranko, V., Montanari, A., Sala, P.: Tableau systems for logics of subinterval structures over dense orderings. In: Olivetti, N. (ed.) TABLEAUX 2007. LNCS (LNAI), vol. 4548, pp. 73–89. Springer, Heidelberg (2007)
3. Bresolin, D., Goranko, V., Montanari, A., Sciavicco, G.: On Decidability and Expressiveness of Propositional Interval Neighborhood Logics. In: Artemov, S.N., Nerode, A. (eds.) LFCS 2007. LNCS, vol. 4514, pp. 84–99. Springer, Heidelberg (2007)
4. Bresolin, D., Montanari, A., Sala, P.: An optimal tableau-based decision algorithm for Propositional Neighborhood Logic. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 549–560. Springer, Heidelberg (2007)
5. Bresolin, D., Montanari, A., Sciavicco, G.: An optimal decision procedure for Right Propositional Neighborhood Logic. Journal of Automated Reasoning 38(1-3), 173–199 (2007)
6. Chaochen, Z., Hansen, M.R.: An adequate first order interval logic. In: de Roever, W.-P., Langmaack, H., Pnueli, A. (eds.) COMPOS 1997. LNCS, vol. 1536, pp. 584–608. Springer, Heidelberg (1998)
7. Goranko, V., Montanari, A., Sciavicco, G., Sala, P.: A general tableau method for propositional interval temporal logics: Theory and implementation. Journal of Applied Logic 4(3), 305–330 (2006)
8. Halpern, J.Y., Shoham, Y.: A propositional modal logic of time intervals. Journal of the ACM 38(4), 935–962 (1991)

9. Moszkowski, B.: Reasoning about digital circuits. Tech. Rep. 83-970, Dept. of Computer Science, Stanford University, Stanford, CA (1983)
10. Otto, M.: Two variable first-order logic over ordered domains. Journal of Symbolic Logic 66(2), 685–702 (2001)
11. Venema, Y.: A modal logic for chopping intervals. Journal of Logic and Computation 1(4), 453–476 (1991)

Normal Form Nested Programs

Annamaria Bria, Wolfgang Faber, and Nicola Leone

Department of Mathematics, University of Calabria, 87036 Rende (CS), Italy
{a.bria,faber,leone}@mat.unical.it

Abstract. Disjunctive logic programming under the answer set semantics (*DLP*, *ASP*) has been acknowledged as a versatile formalism for knowledge representation and reasoning during the last decade. Lifschitz, Tang, and Turner have introduced an extended language of *DLP*, called Nested Logic Programming (*NLP*), in 1999 [1]. It often allows for more concise representations by permitting a richer syntax in rule heads and bodies. However, that language is propositional and thus does not allow for variables, one of the strengths of *DLP*.

In this paper, we introduce a language similar to *NLP*, called Normal Form Nested (*NFN*) programs, which does allow for variables, and present the syntax and semantics. With the presence of variables, domain independence is no longer guaranteed. We study this issue in depth and define the class of safe *NFN* programs, which are guaranteed to be domain independent. Moreover, we show that for *NFN* programs which are also *NLP*s, our semantics coincides with the one of [1]; while keeping the standard meaning of answer sets on DLP programs with variables. Finally, we provide an algorithm which translates *NFN* programs into *DLP* programs, and does so in an efficient way, allowing for the effective implementation of the *NFN* language on top of existing DLP systems.

1 Introduction

In disjunctive logic programming (*DLP*) the heads (resp. the bodies) of rules are disjunctions (resp. conjunctions) of simple constructs, viz. atoms and literals. *DLP*, under the answer set semantics [2,3], is widely recognized as an important tool for knowledge representation and reasoning [4].

Lifschitz, Tang and Turner [1] extended the answer set semantics (in the propositional or ground case) to a class of logic programs where the heads and the bodies of rules are nested expressions. Nested expressions are formed from negation-as-failure literals, conjunction and disjunction, nested arbitrarily. This class of programs, called *nested logic programs*, generalizes the class of disjunctive logic programs. Moreover, as shown in [1,5], nested logic programs can be transformed into disjunctive logic programs. These results allow for evaluating ground nested logic programs using DLP systems, such as DLV [6], GnT [7], or Cmodels3 [8]. However, given that these transformation systems work only for ground nested logic programs, means that variables, one of the strongest features of logic programming, cannot be used in problem representations. This restriction limits the suitability of nested logic programs in many application domains, especially when reasoning is to be done on large numbers of input facts.

Unfortunately, a generalization of these techniques to programs with variables is not straightforward. A major obstacle is the requirement of *DLP* systems that *DLP* rules

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 76–88, 2008.

must be safe, that is each variable in a rule must occur in a positive body literal. When one would just add variables to the method of [5], one easily obtains unsafe rules, as explained in detail in Section 4.

Motivated by these considerations, we extend nonground *DLP* to a class of programs, in which rule heads are formulas in disjunctive normal form made of atoms, and in which the rule bodies are formulas in conjunctive normal form made of literals. These programs are referred to as Normal Form Nested (*NFN*) programs, and are different to nested logic programs of [1], since they may contain variables. We study semantic and safety properties of this class of programs, and provide a polynomial translation from *NFN* programs to *DLP*.

The need for extending DLP with conjunction in the heads and disjunction in the body arises quite often in real world applications. For example, we met the following example in a real-world data-integration application.

Consider a global relation $p(ID, name, surname, age)$ (for persons) with a key-constraint on the first attribute ID. To perform consistent query answering [9], when two tuples share the same key, the relation person is "repaired" by intensionally deleting one of them. In DLP, this is obtained by the following rules (where $\overline{p}$ stands for deleted tuples, and p' is the resulting consistent relation on which query answers are computed).

$$\overline{p}(I,N,S,A) \vee \overline{p}(I,M,T,B) \text{ :- } p(I,N,S,A), p(I,M,T,B), N \neq M.$$
$$\overline{p}(I,N,S,A) \vee \overline{p}(I,M,T,B) \text{ :- } p(I,N,S,A), p(I,M,T,B), S \neq T.$$
$$\overline{p}(I,N,S,A) \vee \overline{p}(I,M,T,B) \text{ :- } p(I,N,S,A), p(I,M,T,B), A \neq B.$$
$$p'(I,N,S,A) \text{ :- } p(I,N,S,A), \textbf{not } \overline{p}(I,N,S,A).$$

The first three DLP rules can be equivalently encoded by a single *NFN* rule, which is much more succinct and intuitive:

$$\overline{p}(I,N,S,A) \vee \overline{p}(I,M,T,B) \text{:- } p(I,N,S,A), p(I,M,T,B), (N \neq M \vee S \neq T \vee A \neq B).$$

For a more involved example, we consider the problem co-CERT3COL – a generalization of graph 3-uncolorability, due to I. Stewart [10]. Without going into details, on the left we report a *DLP* encoding as defined in [11], whereas on the right we report an equivalent *NFN* encoding. Rules r_{11} to r_{13} belong to both encodings.

$r_1: \ v(X) \text{ :- } p(X,Y,V).$ $r_2: \ v(Y) \text{ :- } p(X,Y,V).$ $r_3: \ v(X) \text{ :- } n(X,Y,V).$ $r_4: \ v(Y) \text{ :- } n(X,Y,V).$	$r_a: \ v(X), v(Y) \text{ :- } p(X,Y,V) \vee n(X,Y,V).$
$r_5: \ bool(V) \text{ :- } p(X,Y,V).$ $r_6: \ bool(V) \text{ :- } n(X,Y,V).$ $r_7: \ t(V) \vee f(V) \text{ :- } bool(V).$	$r_b: \ t(V) \vee f(V) \text{ :- } p(X,Y,V) \vee n(X,Y,V).$
$r_8: \ c(X,r) \text{ :- } w, v(X).$ $r_9: \ c(X,g) \text{ :- } w, v(X).$ $r_{10}: \ c(X,b) \text{ :- } w, v(X).$	$r_c: \ c(X,r), c(X,g), c(X,b) \text{ :- } w, v(X).$
$r_{11}: \ c(X,r) \vee c(X,g) \vee c(X,b) \text{ :- } v(X).$ $r_{12}: \ w \text{ :- } p(X,Y,V), t(V), c(X,A), c(Y,A).$ $r_{13}: \ w \text{ :- } n(X,Y,V), f(V), c(X,A), c(Y,A).$	

In the *NFN* version we save seven rules and the intermediate predicate *bool*.

The main contributions of this paper are the following:

- We extend *DLP* with variables introducing conjunctions in the head of the rules and disjunctions in the body of the rules, obtaining a new language, *NFN* Programs. We formally define the syntax and semantics of this language.
- We study the properties of the *NFN* programs showing the following results:
  - The answer sets for *NFN* coincide with the answer sets of [1] for Nested Logic Programs, on the common language fragment.
  - The answer sets for *NFN* coincide with the standard answer sets of [3] on (possibly non ground) DLP programs.
  - We provide a definition of safe *NFN* programs. We show that every safe program is domain independent, that is, it has the same answer sets on each universe extending the constants of the program.
- We present an algorithm that transforms normal form nested programs to disjunctive logic programs such that there is a one-to-one correspondence between their answer sets. The translation is efficient and preserves program safety.

2 *NFN* Language

In this section, we formally define the syntax and semantics of the *NFN* language.

2.1 Syntax

A variable or a constant is a term. An atom is $a(t_1, \ldots, t_n)$ where a is a predicate of arity n and $t_1, \ldots, t_n$ are terms. A literal is either a positive literal p or a negative literal **not** p, where p is an atom. A *basic conjunction* is of the form $(l_1, \ldots, l_n)$ where each $l_1, \ldots, l_n$ is a literal; if each $l_1, \ldots, l_n$ is an atom, the basic conjunction is *positive*. A *basic disjunction* is of the form $(k_1 \vee \ldots \vee k_n)$ where each $k_1, \ldots, k_n$ is a literal; if each $k_1, \ldots, k_n$ is an atom, the basic disjunction is *positive*. The parentheses around basic conjunctions and disjunctions may be omitted in unambiguous occurrences. A *(normal form nested) rule* r is of the following form:

$$C_1 \vee \ldots \vee C_n \text{ :- } D_1, \ldots, D_m. \qquad n, m \geq 0$$

where $C_1, \ldots, C_n$ are positive basic conjunctions and $D_1, \ldots, D_m$ are basic disjunctions. The disjunction $C_1 \vee \ldots \vee C_n$ is the *head* of r while the conjunction $D_1, \ldots, D_m$ is the *body* of r. The set of all basic conjunctions appearing in r is denoted by $H(r)$ while the set of all basic disjunctions is denoted by $B(r)$. Moreover, the set of all positive basic disjunctions of r is denoted by $B^+(r)$ and the set of remaining basic disjunctions is denoted by $B^-(r)$ (i.e. $B^-(r) = B(r) \setminus B^+(r)$). A rule is *positive* if $B(r) = B^+(r)$. If all C_i are atoms and all D_j are literals respectively, the rule is called *standard*.

Example 1 (Normal Form Nested Rule). The following is a normal form nested rule:
$a(X) \vee (b(X), c(X))$:- $d(X), (e(X) \vee$ **not** $f(Y)), (d(X) \vee s(Z) \vee f(X))$.
where $d(X)$ and $(d(X) \vee s(Z) \vee f(X))$ are positive basic disjunctions, while $(e(X) \vee$ **not** $f(Y))$ is not, since it contains the negative literal **not** $f(Y)$.

An *NFN* program P is a finite set of rules. P is a *positive program* if all rules of P are positive. P is a *standard program* if all its rules are standard. In the following, we use Σ to denote a general construct (literal, disjunct, conjunct, etc.). We denote by $const(\Sigma)$ the set of constants that appear in a construct Σ and by *vars*(Σ) the set of variables that appear in Σ. A construct Σ is *ground* iff *vars*$(\Sigma) = \emptyset$.

2.2 Semantics

Program Instantiation. Given an *NFN* program P, let U_P be the set of constants appearing in P and B_P be the set of atoms constructible from the predicates of P with constants in U_P. Let $U \supseteq U_P$ be a set of constants and r an *NFN* rule. A *substitution* is a total function σ : *vars*$(r) \mapsto U$ that maps each variable of r to a constant in U, represented also as the set $\{X/c \mid \sigma(X) = c\}$. Given a substitution σ, a *ground instance* of a construct Σ w.r.t. U is $\Sigma\sigma$, that is the application of σ on Σ. The *instantiation* of r, denoted by $Ground(r, U)$, is the set of all ground instances of r w.r.t. U.

Example 2. Let $U = \{1, 2\}$ be a set of constants and r the rule
$(a(X), b(Y)) \vee c(X)$:- $(p(X, Y) \vee q(Y)), q(X)$.
The instantiation $Ground(r, U)$ is the set of the following rules:
$(a(1), b(1)) \vee c(1)$:- $(p(1, 1) \vee q(1)), q(1)$. $(a(1), b(2)) \vee c(1)$:- $(p(1, 2) \vee q(2)), q(1)$.
$(a(2), b(2)) \vee c(2)$:- $(p(2, 2) \vee q(2)), q(2)$. $(a(2), b(1)) \vee c(2)$:- $(p(2, 1) \vee q(1)), q(2)$.

Given a program P, the *instantiation* of P w.r.t. U, denoted by $Ground(P, U)$ is the union of all instantiations of rules in P: $Ground(P, U) = \bigcup_{r \in P} Ground(r, U)$. As a special case, let $Ground(P)$ = $Ground(P, U_P)$.

Interpretation and Models. An *interpretation* I, for an *NFN* program P, is a set of ground atoms $I \subseteq B_P$. A ground atom a is *true* (resp. *false*) w.r.t. I if $a \in I$ (resp. $a \notin I$). A ground negative literal **not** a is *true* (resp. *false*) w.r.t. I if $a \notin I$ (resp. $a \in I$). A ground basic disjunction L is *true* w.r.t. I if at least one literal of L is true w.r.t. I; otherwise L is *false* w.r.t. I. A ground basic conjunction A is *true* w.r.t. I if all atoms of A are true w.r.t. I; otherwise A is *false* w.r.t. I. Similarly, the head of a ground *NFN* rule r is true w.r.t. I if at least one basic conjunction of $H(r)$ is true w.r.t. I, otherwise it is false w.r.t. I. The body of r is true w.r.t. I if all basic disjunctions of $B(r)$ are true w.r.t. I, otherwise it is false w.r.t. I.

A ground *NFN* rule r is *satisfied* w.r.t. I if the head is true w.r.t. I or the body is false w.r.t. I. A program P is *satisfied* w.r.t. I if all rules of $Ground(P)$ are satisfied. In the following, we denote truth and falsity of a construct Σ with $I \models \Sigma$ and $I \not\models \Sigma$ respectively. Given a rule or a program Δ, we also denote the satisfiability of Δ w.r.t. I by $I \models \Delta$, and unsatisfiability of Δ w.r.t. I by $I \not\models \Delta$. A *model* for P is an interpretation M for P such that $M \models P$. A model M for P is *minimal* if no model N for P exists such that $N \subsetneq M$.

Answer Sets. Next we define a reduct for ground *NFN* programs w.r.t. an interpretation. It can be viewed as a generalization of the reduct defined in [12] and a simplification of the one in [1].

Definition 1 (Reduct). *Let P be a ground NFN program and I an interpretation. The reduct of P w.r.t. I, denoted by P^I, is defined as follows: (1) for each $r \in P$ all false literals w.r.t. I are deleted from each basic disjunction of r; (2) all rules s.t. any basic disjunction becomes empty after the application of item 1, are deleted.*

Next we denote by r^I the rule of P^I obtained from $r \in P$ after the deletion described in item 1 of Definition 1 s.t. no basic disjunction of r becomes empty.

Observation 1. *r^I exists iff $I \models B(r)$.*

Indeed, $I \not\models B(r)$ iff a basic disjunction $D \in B(r)$ exists s.t. $I \not\models D$. That is, for each literal $l \in D$, $I \not\models l$. Since all false literals are deleted from $B(r)$, disjunction D becomes empty. Consequently, r^I does not exist.

Example 3. Consider the following *NFN* program P:

$$a. \qquad b. \qquad f \vee (d, e) \text{ :- } (a \vee \textbf{not } c). \qquad p \text{ :- } (\textbf{not } a \vee \textbf{not } b). \qquad g \text{ :- } (b \vee \textbf{not } a).$$

and interpretation $I = \{a, b, f, g\}$, then P^I is the following program:

$$a. \qquad b. \qquad f \vee (d, e) \text{ :- } (a \vee \textbf{not } c). \qquad g \text{ :- } b.$$

Definition 2 (Answer set for *NFN* program). *Given an NFN program P, an interpretation I is an answer set for P iff I is a minimal model for $Ground(P)^I$.*

The set of answer sets for P is denoted by $AS(P)$.

Example 4. In Example 3, I is an answer set for the program P. Indeed, I is a model for P^I and it is simple to check that no subset $J \subsetneq I$ there exists s.t. J satisfies all rules of P^I. The only other answer set of P is $\{a, b, d, e, g\}$.

Looking at Definition 1, similar to the reduct for *DLP* programs defined in [12], all rules with false body are deleted. Furthermore, from rule bodies of the remaining rules all false body literals are deleted. Without the latter deletion, in some cases it is possible to obtain unintuitive answer sets as shown in the following example.

Example 5. Let us consider program $P = \{c \text{ :- } (c \vee \textbf{not } c).\}$ and interpretation $I = \{c\}$. If we just deleted all rules with false body w.r.t. I, the reduct would be again P and I would be an answer set for P. However, using the reduct of Definition 1, we obtain $\{c \text{ :- } c.\}$, of which I is not a minimal model as $\emptyset$ is also a model. Indeed, I is unintuitive as c is only justified by its own truth, and it is also not an answer set according to [1] (cf. Definition 3).

3 Language Properties

In this section we study important properties of *NFN* programs.

3.1 Equivalence to the Semantics of Lifschitz, Tang, and Turner

In order to differentiate the reducts, in the following we denote the reduct of a construct Σ according to the definition in [1] by Σ^{I_L} .

Example 6. Consider P and I of Example 3. P^{I_L} is the following program:

$$a. \qquad b. \qquad f \vee (d, e) \text{ :- } (a \vee \top). \qquad p \text{ :- } (\bot \vee \bot). \qquad g \text{ :- } (b \vee \bot).$$

Definition 3 (Answer Set of Lifschitz, Tang, and Turner [1]). *A set of atoms I is an NLP answer set for a ground NFN program P iff it is a minimal model of P^{I_L}.*

Example 7. In Example 6, I is an *NLP* answer set of the program P^{I_L}.

Lemma 1. *Let r be a ground NFN rule and I a set of atoms, then: (1) $I \not\models B(r) \Leftrightarrow I \not\models B(r^{I_L})$ and (2) if r^I exists, $\forall J \subseteq I: \quad J \models r^I \Leftrightarrow J \models r^{I_L}$.*

Proof. **(1)** $I \not\models B(r)$ iff a basic disjunction $D \in B(r)$ exists s.t. $I \not\models D$ iff for all $l \in D$, $I \not\models l$. The corresponding basic disjunction $D^{I_L} \in B(r^{I_L})$ of D contains the positive literals of D and, if $l =$ **not** a, D^{I_L} contains $\bot$. Then $I \not\models B(r^{I_L})$. Vice versa, $I \not\models B(r^{I_L})$ iff $D^{I_L} \in B(r^{I_L})$ exists s.t. $I \not\models D^{I_L}$ iff for each $l \in D^{I_L}$, $I \not\models l$ where l is an atom or $l = \bot$. The corresponding basic disjunction $D \in B(r)$ contains the atoms of D^{I_L} and, for each $\bot \in D^{I_L}$, D contains a negative literal l_n, s.t. $I \not\models l_n$. Consequently, $I \not\models D$ and, hence, $I \not\models B(r)$.

(2) $J \models B(r^I)$ iff for each basic disjunction $D^I \in B(r^I)$ a literal $l \in D^I$ exists s.t. $J \models l$. If $l =$ **not** a, $I \models l$ follows from Definition 1 and each basic disjunction $D^{I_L} \in B(r^{I_L})$ corresponding to D^I contains $\top$. Otherwise, if l is a positive literal, the D^{I_L} corresponding to D^I also contains l. As a result $J \models B(r^{I_L})$. Since both reducts do not modify the head of the rules, if $J \models r^I$ and $J \models B(r^I)$ then $J \models H(r^I) = H(r^{I_L})$ and $J \models B(r^{I_L})$ hence $J \models r^{I_L}$; if $J \models r^{I_L}$ and $J \models B(r^{I_L})$ then $J \models H(r^{I_L}) = H(r^I)$ and $J \models B(r^I)$, hence $J \models r^I$.

Theorem 1. *Given an NFN program P, an interpretation I is an answer set of P according to Definition 2 iff I is an NLP answer set of P according to Definition 3.*

Proof. ($\Rightarrow$) If I is a minimal model for $Ground(P)^I$, for each $r \in Ground(P)$, s.t. r^I exists, from Observation 1 $I \models B(r)$ and from (2) of Lemma 1 (for the special case $I = J$), $I \models r^{I_L}$ follows. For rules $r \in Ground(P)$, for which no r^I but a r^{I_L} exists, $I \not\models r$ and from (1) of Lemma 1, $I \not\models B(r^{I_L})$. Consequently I is a model for $Ground(P)^{I_L}$. Moreover, no $J \subset I$ is a model for $Ground(P)^{I_L}$, as it would also be a model for $Ground(P)^I$ because of (2) of Lemma 1.

($\Leftarrow$) Let I be a minimal model for $Ground(P)^{I_L}$. For each $r^I \in Ground(P)^I$, since $I \models r^{I_L}$ holds by (2) of Lemma 1, $I \models r^I$ holds as well. As a result, I is a model for $Ground(P)^I$. Furthermore, no $J \subset I$ is a model for $Ground(P)^I$ because J would also be a model for $Ground(P)^{I_L}$. In fact, for each $r^I \in Ground(P)^I$ from (2) of Lemma 1, $J \models r^{I_L}$. For each $r \in Ground(P)$ s.t. no r^I exists, from Observation 1 $I \not\models B(r)$, therefore a disjunction $D \in B(r)$ exists s.t. $I \not\models D$ iff $I \not\models l$ for all $l \in D$. The corresponding disjunction $D^{I_L} \in B(r^{I_L})$ contains the same atoms of D and D^{I_L} contains $\bot$ for each negative literal of D. Consequently $J \not\models D^{I_L}$ for all $J \subseteq I$ therefore $J \not\models B(r^{I_L})$ and $J \models r^{I_L}$.

Since the grounding of a standard program defined in this paper is the same as in [3] we obtain, by virtue of Theorem 1 and results of [1], the following.

Proposition 1. *Given a standard DLP program P, the answer sets of P according to Definition 2 coincide with the answer sets defined by Gelfond and Lifschitz in [3].*

3.2 Domain Independence and Safety

Let us review the definition of domain independence, stating in our case that the semantics should be independent of the universe, as long as it is sufficiently large.

Definition 4. *Let P be an NFN program and U_P be the set of constants appearing in P. P is* domain independent *if for each $U \supseteq U_P$, $AS(Ground(P,U)) = AS(Ground(P,U_P))$ holds.*

Let us examine some examples related to domain independence.

Example 8. Consider $P_{us} = \{c(1)., d(1)., a(X) \vee b(Y)$:- $(c(X) \vee d(Y)).\}$ where $U_{P_{us}} = \{1\}$. Then $Ground(P_{us}, U_{P_{us}}) = \{c(1)., d(1)., a(1) \vee b(1)$:- $(c(1) \vee d(1)).\}$. The answer sets are: $AS(Ground(P_{us}, U_{P_{us}})) = \{\{a(1), c(1), d(1)\}, \{b(1), c(1), d(1)\}\}$. Now we consider $U = U_{P_{us}} \cup \{2\}$. Then $Ground(P_{us}, U) = Ground(P_{us}, U_{P_{us}}) \cup \{a(1) \vee b(2)$:- $(c(1) \vee d(2))., a(2) \vee b(1)$:- $(c(2) \vee d(1))., a(2) \vee b(2)$:- $(c(2) \vee d(2)).\}$. In this case we have the following answer sets: $AS(Ground(P_{us}, U)) = \{\{a(1), a(2), c(1), d(1)\}, \{a(1), b(1), c(1), d(1)\}, \{b(1), b(2), c(1), d(1)\}\}$. So we obtain different answer sets for the program P_{us}, depending on the considered universe. The problem is that in the main rule of P_{us} the body can be satisfied without "binding" one of the two variables that occur in the head.

Example 9. Consider $P_{us2} = \{c(1)., a$:- $(b(X) \vee$ **not** $c(X)).\}$ where $U_{P_{us2}} = \{1\}$. Then $Ground(P_{us2}, U_{P_{us2}}) = \{c(1)., a$:- $(b(1) \vee$ **not** $c(1)).\}$, so the program has the only one answer set: $AS(Ground(P_{us2}, U_{P_{us2}})) = \{c(1)\}$. Now if $U_2 = U_{P_{us2}} \cup \{2\}$, $Ground(P_{us2}, U_2) = Ground(P_{us2}, U_{P_{us2}}) \cup \{a$:- $(b(2) \vee$ **not** $c(2)).\}$ So we have the following answer set: $AS(Ground(P_{us2}, U_2) = \{a, c(1)\}$. In this case the variable in the negative literal is not necessarily "bound" when the body is true.

These examples serves as a proof for the following theorem.

Theorem 2. *Given an NFN program P, P is in general not domain independent.*

Definition 5 (Safe variable). *Let r be an NFN rule. A variable $X \in$ vars(r) is* safe *if there exists a positive basic disjunction $D \in B(r)$, such that $\forall a \in D$, $X \in$ vars(a); we also say that D saves X and X is made safe by D.*

Example 10. Consider a :- $(b(X) \vee c(X,Z) \vee d(X)), e(Y), (s(Z) \vee t(X))$. The safe variables of the rule are X and Y. Indeed, the variable X is safe because it appears in all atoms of the positive basic disjunction $D_1 = (b(X) \vee c(X,Z) \vee d(X))$, while the variable Y occurs in the only atom of the positive basic disjunction $e(Y)$.

Definition 6 (Safe rules and programs). *An NFN rule r is safe if each variable that appears in the head of r and each variable that appears in negative body literals of r is safe. An NFN program P is safe if each rule is safe.*

Example 11. The *NFN* rule h :- $(a(X) \vee b(X))$, **not** $c(X)$. is safe. In fact, variable X, which appears in the negative literal **not** $c(X)$ is made safe by $(a(X) \vee b(X))$. Rule $(h_1(X), h_2(X))$:- $(a(X) \vee b(Z)), (c(X) \vee$ **not** $s(Z))$. is not safe, since variable X occurs in the head of the rule but no positive basic disjunction in the body saves X. Moreover, variable Z, occurring in negative body literal **not** $s(Z)$, is also unsafe.

Lemma 2. *Let r be a safe NFN rule, I a set of ground atom, $U \supseteq const(I)$ and $U' \supset U$, then $I \models Ground(r,U) \Rightarrow I \models Ground(r,U')$.*

Proof. Assume $I \not\models Ground(r,U')$ and $I \models Ground(r,U)$. Then, a substitution $\sigma : vars(r) \rightarrow U'$ exists s.t. $I \not\models r\sigma$, so $I \models B(r\sigma)$ *and* $I \not\models H(r\sigma)$. Since $I \models B(r\sigma)$, then for each positive basic disjunction $D_p \in B^+(r)$ an atom $a \in D_p$ exists s.t. $a\sigma \in I$ so $const(a\sigma) \subseteq U$. Moreover, for each negative basic disjunction $D_n \in B^-(r)$ a literal $l \in D_n$ exists s.t. $I \models l\sigma$. Therefore, if l is positive, $l\sigma \in I$ and $const(l\sigma) \subseteq U$; if $l =$ **not** a, $a\sigma \notin I$ and since r is safe, for all $X \in vars(a)$ a positive disjunction $D_s \in B^+(r)$ exists s.t. for all $\bar{a} \in D_s$, $X \in vars(\bar{a})$ and thus $const(l\sigma) \subseteq U$. Then, choose a substitution $\sigma' : vars(r) \mapsto U$ such that $\forall\, X/c \in \sigma$ s.t. $c \in (U' \setminus U)\ \exists s \in U$ s.t. $X/s \in \sigma'$, and $\forall\, X/c \in \sigma$ s.t. $c \in U$, $X/c \in \sigma'$. Then it holds that $I \models B(r\sigma')$ and $I \models H(r\sigma')$ because $r\sigma' \in Ground(P,U)$. Furthermore, r is safe so for all $X \in vars(H(r))$ a positive disjunction $D_p \in B^+(r)$ exists s.t. for all $\bar{a} \in D_p$, $X \in vars(\bar{a})$ and thus $const(H(r\sigma)) \subseteq U$, hence $H(r\sigma') = H(r\sigma)$, and we obtain a contradiction.

Lemma 3. *Let r be a safe NFN rule, I a set of ground atoms, and $U \supseteq const(I)$, then $Ground(r,U)^I = Ground(r,U')^I$ for all $U' \supset U$.*

The proof is straightforward and therefore omitted.

Theorem 3. *If P is safe then P is domain independent.*

Proof. $\forall U \supseteq U_P : AS(Ground(P,U_P)) \subseteq AS(Ground(P,U))$:
Let P be a safe program and assume that $I \in AS(Ground(P,U_P))$ and $\exists U \supset U_P$ s.t. $I \notin AS(Ground(P,U))$. (i) If I is not a model of $Ground(P,U) \Rightarrow \exists r \in P$ and a substitution $\sigma : vars(r) \mapsto U$ s.t. $I \not\models r\sigma$. Since $I \models Ground(P,U_P) = \bigcup_{r' \in P} Ground(r',U_P)$, for each $r' \in P$ $I \models Ground(r',U_P)$ and from Lemma 2, $I \models Ground(r',U)$. Then $I \models r\sigma$ for each $r\sigma \in Ground(P,U)$ and we obtain a contradiction. (ii) If I is a model for $Ground(P,U)$ but I is not a minimal model for $Ground(P,U)^I$, then $\exists J \subset I$ s.t. J is a model for $Ground(P,U)^I$. Since $Ground(P,U_P) \subset Ground(P,U)$, then $(Ground(P,U_P)^I) \subseteq (Ground(P,U)^I)$ and J is a model for $Ground(P,U_P)^I$ contradicting $I \in AS(Ground(P,U_P))$.

$\forall U \supseteq U_P : AS(Ground(P,U)) \subseteq AS(Ground(P,U_P))$: Let P be a safe program and assume that $I \in Ground(P,U)$ and $\exists U \supset U_P$ s.t. $I \notin AS(Ground(P,U_P))$. (i) Since $Ground(P,U_P) \subset Ground(P,U)$ then I is a model for $Ground(P,U_P)$. (ii) If I is a model for $Ground(P,U_P)$ but I is not a minimal model for $Ground(P,U_P)^I$ then there exists $J \subset I$ s. t. $J \in AS(Ground(P,U_P)^I)$. By Lemma 3 it holds that $Ground(P,U_P)^I = Ground(P,U)^I$, and therefore $J \models Ground(P,U)^I$ and this is a contradiction to $I \in AS(Ground(P,U))$.

4 An Efficient Translation from *NFN* to *DLP*

Results in [1] show that ground nested logic programs can be transformed into ground standard disjunctive logic programs. This allows for evaluating nested logic programs using a disjunctive logic programming system, such as DLV [6], GnT [7], or Cmodels3

[8], as a back-end. However, the naïve transformation will in general produce very large programs, which may be exponentially larger than the input program. Inspired by structure-preserving normal form translations [13], a much more efficient transformation has been described in [5], which is guaranteed to run in polynomial time and will also provide transformed programs of polynomial size with respect to the input program. The main idea of this transformation is to introduce labels or auxiliary atoms that represent subformulas. However, also the latter transformation works for ground programs only. In this section we present an algorithm which efficiently translates normal form nested programs, which may contain variables, to disjunctive logic programs, elaborating on some ideas of [5].

Let us first observe that a naïve transformation on NFN programs amounts to transforming CNFs in DNFs and vice versa, so also for NFN programs it may create a program which is exponentially larger than the original. For example, a rule of the form h :- $(a_1 \vee b_1), \ldots, (a_n \vee b_n)$. would generate 2^n transformed rules. So also for NFN programs, the introduction of auxiliary atoms is necessary to avoid this inefficiency.

In what follows, we will use *NFN* program P as running example, where P consists of the following single safe rule:

$$r: \quad a \vee (g(X), h(X,Z)) \text{ :- } (b(X,Z) \vee c(X,Y)), (\textbf{not}\ d(X) \vee f(X,Y)), e(X,Z). \qquad (1)$$

This example shows the problem with a simple-minded lifting of the transformation described in [5]. If one just introduced a new predicate that replaces basic disjunctions in rule bodies and adds all variables of the basic disjunction that also occur elsewhere in the rule, the defining rules for the new predicate would not be safe, but *ASP* system require safe input. For instance, if the disjunction $(b(X,Z) \vee c(X,Y))$ of the running example is to be replaced by an auxiliary predicate l, we would need two defining rules for l, $l(X,Y,Z)$:- $b(X,Z)$. and $l(X,Y,Z)$:- $c(X,Y)$. Both of these rules are unsafe. However, the unsafe variables Y and Z must in some way be included in the atom for l, as in the rewritten rule other occurrences of Y and Z would be decoupled from the occurrences in the rewritten disjunction. Note that the variable Z is safe in rule (1), while Y is not. These kinds of variable occurrences are the main issues to be solved in the rewriting, so let us formally define them.

Definition 7 (Shared and Unrestricted Variables). *Let D be a basic disjunction of a safe NFN rule r. A variable $X \in \text{vars}(D)$ is shared in D and r if X is a safe variable of r but X is not made safe by D. A variable $X \in \text{vars}(D)$ is unrestricted in D and r if X is not a safe variable of r and there exists a disjunction $D' \neq D$ in r s.t. $X \in \text{vars}(D')$.*

We denote the set of shared variables of a basic disjunction D by $Shared_D$ and by $Unres_D$ the set of unrestricted variables of D. Note that the presence of unrestricted variables does not imply that the rule is unsafe.

Example 12. Consider rule (1) where $D_1 = (b(X,Z) \vee c(X,Y))$, and $D_2 = (\textbf{not}\ d(X) \vee f(X,Y))$. $Shared_{D_1} = \{Z\}$, $Unres_{D_1} = \{Y\}$, $Shared_{D_2} = \{X\}$, $Unres_{D_2} = \{Y\}$.

Let us now review the rewriting Algorithm *rewriteNFN*, shown in Fig. 1. Its input is a safe *NFN* program P and it returns a safe standard DLP program, P_{DLP}. The algorithm transforms one rule at a time, creating one major rule, which directly represents the original one, and possibly several auxiliary rules. Moreover, all rules built by the algorithm are safe.

begin *rewriteNFN*
Input: NFN program P
var B: conjunction of literals; H: disjunction of atoms; $auxh^r_C$, aux^r_D: atoms;
Output: DLP program P_{DLP}.
  $P_{DLP} := \emptyset$;
  for each rule $r \in P$ **do**
    $H := \epsilon$; $B := \epsilon$; (initialization)
    for each basic conjunction $C \in H(r)$ **do**
      buildAuxiliaryHeadAtom$(C, auxh^r_C, P_{DLP})$; (see Section 4.1)
      $H := H \vee auxh^r_C$;
    for each basic disjunction $D \in B(r)$ **do**
      buildAuxiliaryBodyAtom(D, aux^r_D, P_{DLP}); (see Section 4.2)
      $B := B, aux^r_D$;
    addAtomsForMatching(B, P_{DLP}); (see Section 4.3)
    $P_{DLP} := P_{DLP} \cup \{H$:- $B.\}$;
  end for
  return P_{DLP};
end.

Fig. 1. Algorithm: *rewriteNFN*

4.1 Head Transformation

The rewriting of the rule head is a fairly direct lifting of the respective transformation of [5]. Obviously, in our case we have to take care of variables, and as an optimization, we do not introduce auxiliary atoms for conjunctions consisting of a single atom. Function *buildAuxiliaryHeadAtom*, given basic conjunction C, returns an auxiliary atom with predicate name $auxh^r_C$ and all variables in C. The function also adds the rule $auxh^r_C$:- C. and for each $a_i \in C$ the rule a_i :- $auxh^r_C$. to P_{DLP}.

Example 13. In the rewritten major rule, the basic conjunction $(g(X), h(X, Z))$ of rule (11) is replaced by $auxh^r_1(X, Z)$, and the following rules are added to P_{DLP}:
$auxh^r_1(X, Z)$:- $g(X), h(X, Z)$. $h(X, Z)$:- $auxh^r_1(X, Z)$. $g(X)$:- $auxh^r_1(X, Z)$.

4.2 Body Transformation

As for the rule body, the basic idea is to transform one basic disjunction at a time, keeping single-literal disjunctions, and introducing an auxiliary atom with predicate aux^r_D for each disjunction D containing more than one literal. The definition of this predicate will contain a rule for each disjunct of D. However, as motivated earlier, care has to be taken with variables. In particular, we have to find a solution for dealing with variables that are not made safe by the disjunction to be rewritten, but still occur somewhere else in the rule body. There must be an argument for those variables in the auxiliary atom, but some literals of the disjunction may not contain this variable, leaving the respective defining rule of aux^r_D unsafe.

Our solution, implemented in function *buildAuxiliaryBodyAtom*, is to introduce a new constant $\#u$, which represents the fact that the respective variable is not bound. In order to work correctly, however, $\#u$ must match with all other constants. We achieve this by defining a predicate $match^r_X$ for a variable X which extends standard matching by these special matching rules for $\#u$. In order to restrict the value ranges for these predicates, we also define predicates u^r_X, which holds for all constants that X may possibly become bound to.

For the definition of the *match* predicates, we differentiate between shared and unrestricted variables. Shared variables are made safe by some other disjunction in the body, so these variables are guaranteed to be bound to a constant value. In this case, it is sufficient to match a value which is possibly unrestricted to a value which is definitely bound (originating from a saving disjunction). The *match* predicate in this case needs only two arguments. For unrestricted variables, the presence of a bound value is not guaranteed, and in this case we need a *match* predicate with three arguments: The third argument indicates whether the other two arguments are unrestricted, or whether a bound value has been found; see Example 14 for how this is achieved.

Example 14. For the first basic disjunction of rule (1) the auxiliary atom created by *buildAuxiliaryBodyAtom* is $aux^r_1(X, Y, Z)$. The rules added to P_{DLP} are

$aux^r_1(X, \#u, Z)$:- $b(X, Z)$. $aux^r_1(X, Y, \#u)$:- $c(X, Y)$.
$match^r_Y(\#u, \#u, \#u)$. $match^r_Y(Y, Y, Y)$:- $u^r_Y(Y)$.
$match^r_Y(Y, \#u, Y)$:- $u^r_Y(Y)$. $match^r_Y(\#u, Y, Y)$:- $u^r_Y(Y)$.
$match^r_Z(Z, Z)$:- $u^r_Z(Z)$. $match^r_Z(\#u, Z)$:- $u^r_Z(Z)$.
$u^r_Y(Y)$:- $c(X, Y)$.

Note that the definition of u^r_Y is not complete yet; when processing the second basic disjunction, the rule $u^r_Y(Y)$:- $f(X, Y)$. will be added. In a similar way, the definition of u^r_Z does not exist yet; since Z is a safe variable in the rule, the universe for Z is sufficiently defined by one disjunction that saves Z. In our example, eventually rule $u^r_Z(Z)$:- $e(X, Z)$. will be added.

There is another issue to resolve for negative literals, as also variables occurring in negative literals will render the respective defining rule for aux^r_D unsafe, even if the technique using $\#u$ is used to eliminate the variable from the head. However, since the original rule is safe, such variables are guaranteed to be bound to a constant value. This allows us to add a definition for the values that this variable can assume. We use the predicate sav^r_X for a variable X to this end, and the defining rules will contain all atoms that represent the disjunctions of the original rule that save X.

Example 15. For the second basic disjunction of rule (1) the auxiliary atom created by *buildAuxiliaryBodyAtom* is $aux^r_2(Y)$. X, a shared variable, is not included in the arguments just for optimization (if no variable is saved by a disjunction, shared variables will not be included as arguments). The rules added to P_{DLP} are

$aux^r_2(\#u)$:- **not** $d(X), sav^r_X(X)$. $aux^r_2(Y)$:- $f(X, Y), sav^r_X(X)$. $u^r_Y(Y)$:- $f(X, Y)$.

Here $sav^r_X(X)$ is added to both rule bodies because of the optimization mentioned before. The rule defining sav^r_X is composed of all atoms representing disjunctions that save X, taking care to match shared variables (and also unrestricted ones, which however is not applicable in our example).

4.3 Composing the Major Rule

What remains to be done is to create the major rewritten rule. The rule head and the main part of the body have already been created. What is still missing is adding atoms for the extended match, which also takes care of the special constant $\#u$. To this end, function *addAtomsForMatching* first renames all shared and unrestricted variable occurrences in the atoms of the preliminary major rule body, using new names (denoted here by adding an index to the original variable). It then adds an atom $match^r_X(X_i, X)$ to the major rule body for each shared variable occurrence that has been renamed to X_i. Furthermore, for each unresticted variable X, we work along each renamed occurrence from left to right. If the occurrences are $X_1, X_2, \ldots, X_n$, we add an atom $match^r_X(X_1, X_2, X_{\leq 2})$, then $match^r_X(X_{\leq i}, X_{i+1}, X_{\leq i+1})$ for each $2 \leq i < n$.

Example 16. For rule (11) the major rule built so far is $a \vee auxh^r_1(X,Z)$:- $aux^r_1(X,Y,Z)$, $aux^r_2(Y), e(X,Z)$. which *addAtomsForMatching* transforms to $a \vee auxh^r_1(X,Z)$:- $aux^r_1(X,Y_1,Z_1), aux^r_2(Y_2), e(X,Z), match^r_Y(Y_1,Y_2,Y_{\leq 2}), match^r_Z(Z_1,Z)$.

As a final step, we add a defining rule for each sav^r_X that has been created. The rule body is composed of all atoms representing saving disjunctions of X (at least one must exist since X is safe). Then, we apply the same procedure as for the major rule in order to rename shared and unrestricted variables and add respective match atoms.

Example 17. For rule (11) we add $sav^r_X(X)$:- $aux^r_1(X,Y,Z_1), e(X,Z), match^r_Z(Z_1,Z)$.

Summarizing, for the running example, *rewriteNFN* creates:

$a \vee auxh^r_1(X,Z)$:- $aux^r_1(X,Y_1,Z_1), aux^r_2(Y_2), e(X,Z), match^r_Y(Y_1,Y_2,Y_{\leq 2}), match^r_Z(Z_1,Z)$.
$sav^r_X(X)$:- $aux^r_1(X,Y,Z_1), e(X,Z), match^r_Z(Z_1,Z)$.
$auxh^r_1(X,Z)$:- $g(X), h(X,Z)$. $h(X,Z)$:- $auxh^r_1(X,Z)$. $g(X)$:- $auxh^r_1(X,Z)$.
$aux^r_1(X,\#u,Z)$:- $b(X,Z)$. $aux^r_1(X,Y,\#u)$:- $c(X,Y)$.
$aux^r_2(Y)$:- $f(X,Y), sav^r_X(X)$. $aux^r_2(\#u)$:- **not** $d(X), sav^r_X(X)$.
$match^r_Y(Y,\#u,Y)$:- $u^r_Y(Y)$. $match^r_Y(\#u,Y,Y)$:- $u^r_Y(Y)$. $match^r_Y(\#u,\#u,\#u)$.
$match^r_Y(Y,Y,Y)$:- $u^r_Y(Y)$. $match^r_Z(\#u,Z)$:- $u^r_Z(Z)$. $match^r_Z(Z,Z)$:- $u^r_Z(Z)$.
$u^r_Y(Y)$:- $c(X,Y)$. $u^r_Y(Y)$:- $f(X,Y)$. $u^r_Z(Z)$:- $e(X,Z)$.

Consider now $F = \{c(1,1)., e(1,1).\}$, $P' = P \cup F$, and $P_{DLP}{}' = P_{DLP} \cup F$. The answer sets of P' are $A_1 = \{a, c(1,1), e(1,1)\}$ and $A_2 = \{g(1), h(1,1), c(1,1), e(1,1)\}$. while the answer sets of P'_{DLP} are $A'_1 = \{\mathbf{a}, \mathbf{c(1,1)}, \mathbf{e(1,1)}, aux^r_1(1,1,\#u), aux^r_2(\#u), u^r_Y(1), u^r_Z(1), sav^r_X(1), match^r_Z(1,1), match^r_Z(\#u,1), match^r_Y(\#u,1,1), match^r_Y(1,\#u,1), match^r_Y(1,1,1), match^r_Y(\#u,\#u,\#u)\}$ and $A'_2 = \{\mathbf{g(1)}, \mathbf{h(1,1)}, \mathbf{c(1,1)}, \mathbf{e(1,1)}, auxh^r_C(1,1), aux^r_1(1,1,\#u), aux^r_2(\#u), u^r_Y(1), u^r_Z(1), sav^r_X(1), match^r_Z(1,1), match^r_Z(\#u,1), match^r_Y(\#u,1,1), match^r_Y(1,\#u,1), match^r_Y(1,1,1), match^r_Y(\#u,\#u,\#u)\}$

Note that there is a one-to-one correspondence between the answer sets of P' and P'_{DLP}. We can prove that such a correspondence holds in any case.

Proposition 2. *Let P a safe NFN program, P_{DLP} = rewriteNFN(P), and $\mathcal{A}_N$ and $\mathcal{A}_D$ be the sets of predicate symbols that appear in P and in P_{DLP}, respectively ($\mathcal{A}_N \subseteq \mathcal{A}_D$). Then, $I \in AS(P)$ iff there exists a unique $J \in AS(P_{DLP})$ s.t. $I = J \cap \mathcal{A}_N$.*

As mentioned previously, all rules generated by *rewriteNFN* are safe. Moreover, it is not hard to see that the complexity of the algorithm is lightweight.

Proposition 3. *Let P be a safe NFN program, then* rewriteNFN(P) *is a safe DLP program, its size is polynomial in $|P|$, and it is computed in polynomial time in $|P|$.*

5 Conclusion

We have introduced *NFN* programs, an extension of nonground disjunctive logic programs, where conjunctions of atoms and disjunctions of literals are permitted in the heads and in the bodies of the rules, respectively. We have defined syntax and semantics of the new language and, since ground *NFN* programs are *NLP* programs, we showed that the respective notions of answer sets coincide on this fragment. Furthermore, we have defined the class of safe *NFN* programs and showed that each program of this class is domain independent, that is, has the same answer sets for each universe containing the constants of the program. Finally, we have developed an algorithm that rewrites an *NFN* program P into a *DLP* program P_{DLP}. The size of P_{DLP} is polynomial in $|P|$, the algorithm runs in polynomial time and preserves program safety. Ongoing work concerns the implementation of the presented algorithm, allowing for the computation of answer sets for *NFN* programs by exploiting disjunctive ASP systems as back-ends.

References

1. Lifschitz, V., Tang, L.R., Turner, H.: Nested Expressions in Logic Programs. AMAI 25(3–4), 369–389 (1999)
2. Przymusinski, T.C.: Stable Semantics for Disjunctive Programs. NGC 9, 401–424 (1991)
3. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. NGC 9, 365–385 (1991)
4. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, Cambridge (2003)
5. Pearce, D., Sarsakov, V., Schaub, T., Tompits, H., Woltran, S.: A Polynomial Translation of Logic Programs with Nested Expressions into Disjunctive Logic Programs: Preliminary Report. In: Stuckey, P.J. (ed.) ICLP 2002. LNCS, vol. 2401, pp. 405–420. Springer, Heidelberg (2002)
6. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. ACM TOCL 7(3), 499–562 (2006)
7. Janhunen, T., Niemelä, I.: Gnt - a solver for disjunctive logic programs. In: Lifschitz, V., Niemelä, I. (eds.) LPNMR 2004. LNCS (LNAI), vol. 2923, pp. 331–335. Springer, Heidelberg (2003)
8. Lierler, Y.: Disjunctive Answer Set Programming via Satisfiability. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 447–451. Springer, Heidelberg (2005)
9. Bertossi, L.E.: Consistent query answering in databases. SIGMOD Record 35(2), 68–76 (2006)
10. Stewart, I.A.: Complete problems involving boolean labelled structures and projection transactions. Journal of Logic and Computation 1(6), 861–882 (1991)
11. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive Datalog. ACM TODS 22(3), 364–418 (1997)
12. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: Semantics and complexity. In: Alferes, J.J., Leite, J.A. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 200–212. Springer, Heidelberg (2004)
13. Plaisted, D.A., Greenbaum, S.: A Structure-Preserving Clause Form Translation. Journal of Symbolic Computation 2(3), 293–304 (1986)

A Logic for Closed-World Interaction

Jan Broersen, Rosja Mastop, John-Jules Ch. Meyer, and Paolo Turrini

Universiteit Utrecht

Abstract. The aim of the work is to provide a language to reason about closed-world interaction, that is all those situations in which the outcomes of an interaction can be determined by the agents themselves and in which Nature does not play an active role. We formalize this intuition by identifying all such interactions and axiomatizing their logic. We apply the formal tools to reason about games and their regulation.

1 Introduction

Pauly's Coalition Logic has shown to be a sound formal tool to analyze the properties of strategic interactions and games. One issue left is to define in that language what the interesting properties of an interaction are, as possible for instance with regularity (it is never the case that a group of agents can determine that some variable p is true, while all the other agents can at the same time determine that p is false) or outcome monotonicity (if a coalition can force an outcome to lie in a set X, can also force an outcome to lie in all supersets of X).

An intuitive property is that of *coherence*, a constrain on the interaction that ensures players' abilities non to contradict one other and the empty coalition not to make active choices. With this property we can model closed-world interaction, such as those of a Coordination Game or of a Prisoner Dilemma, where all the outcomes are determined only by the choices of the agents that are present.

1.1 Motivating Example

Suppose we were confronted with a legislator who wants to regulate an interaction, mandating the optimal outcomes that result from the choices of the coalitions.

Table 1. Clothing Conformity

Row \ Column	White Dress	Black Dress
White Dress	$(3,3)$	$(0,0)$
Black Dress	$(0,0)$	$(3,3)$

Let us consider for instance conventional norms, by which players should conform to each other. In this situation (see Table 1), a legislator that wants to achieve the socially optimal state (players coordinate), should declare that a discordant choice is forbidden,

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 89–99, 2008.

thereby labeling the combinations of moves (black, white), (white, black) as violations. Provided the legislator's aim, we ask ourselves whether it makes sense to construct a regulation for any type of interaction.

Suppose the environment were active part of the game, and it could decide to transform the game of table 1 in the one of table 2.

Table 2. Clothing Conformity Modified

Row \ Column	White Dress
White Dress	$(3, 3)$
Black Dress	$(0, 0)$

What should then the legislator do? It is quite clear that imposing the agents to choose something should depend on the moves that are available to the players. But in a game in which Nature plays an active role, taking this statement serious would boil down to mentioning Nature in the deontic language, saying for instance "Nature should allow Row to play only white" or "Nature should make it convenient for the Grand Coalition to form". No legislator though would be in the condition of determining what moves Nature would play. We take here a view of Nature as an uncontrollable outside, which it is useful to think of as the *initial setting* of the Multi-Agent System. Nature, unlike all the other players, does not have explicit preferences over the outcomes of the interaction and intuitively it does not follow proper man made norms or orders. In order to have a regulation of the Multi Agent System, we need a proper agent-oriented deontic language and we should then avoid deontic statements that concern proper choices to be carried out by Nature. Or put it differently, we need to identify all those interactions for which it makes sense to construct a regulation. This translates into ruling out all those in which Nature plays an active role. In this paper we will pursue this idea formally, identifying all such interactions and axiomatizing their logic.

The paper is structured as follows: In the first part we introduce the notion of *coherence*. We prove that this property cannot be defined in Pauly's Coalition Logic (CL), due to the presence of the inability of the empty coalition (that ensures Nature not to condition the choices of the other coalitions, IOEC henceforth). In the second part we discuss the axiomatization of Coherent Coalition Logic (CCL), an extension of the language of CL, giving a characterization of IOEC in terms of a global modality. In the third part we will discuss some possible applications of the logic to the study and the regulation of closed-world strategic interaction.

2 Coherent Interactions

We define the strategic abilities of agents and coalitions, introducing the concept of a dynamic Effectivity Function (EF), adopted from [6].

Definition 1 (Dynamic Effectivity Function)
Given a finite set of agents Agt and a set of states W, a dynamic Effectivity Function *is a function $E : W \rightarrow (2^{Agt} \rightarrow 2^{2^W})$.*

Any subset of Agt will henceforth be called a *coalition*.

For elements of W we use variables $u, v, w, \ldots$; for subsets of W we use variables $X, Y, Z, \ldots$; and for sets of subsets of W (i.e., elements of 2^{2^W}) we use variables $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \ldots$. The elements of W are called 'states' or 'worlds'; the subsets of Agt are called 'coalitions'; the sets of states $X \in E(w)(C)$ are called the 'choices' of coalition C in state w and this will be abbreviated with wE_CX when needed. The set $E(w)(C)$ is called the 'choice set' of C in w. The complement of a set $\overline{X}$ or of a choice set $\overline{\mathcal{X}}$ are calculated from the obvious domains.

A dynamic Effectivity Function assigns, in each world, to every coalition a set of sets of states. Intuitively, if $X \in E(w)(C)$ the coalition is said to be able to *force* or *determine* the state after w be some member of the set X. If the coalition has this power, it can thus prevent that any state *not* in X will be the next state, but it might not be able to determine *which* state in X will be the next state. Possibly, some other coalition will have the power to refine the choice of C.

To study closed-world interaction we isolate a set of minimally required properties, that constitute the class of *coherent Effectivity Functions*.

Definition 2 (Coherence)
For any world w, coalitions C,D and choice X, an Effectivity Function is coherent *if it has the following properties:*

1. *coalition monotonicity: if $X \in E(w)(C)$ and $C \subseteq D$ then $X \in E(w)(D)$;*
2. *regularity: if $X \in E(w)(C)$ then $\overline{X} \notin E(w)(\overline{C})$;*
3. *outcome monotonicity: if $X \in E(w)(C)$ and $X \subseteq Y$ then $Y \in E(w)(C)$;*
4. *inability of the empty coalition: $E(w)(\emptyset) = \{W\}$.*

The first property says that the ability of a coalition is preserved by enlarging the coalition. In this sense we do not allow new members to interfere with the preexistent capacities of a group of agents. The second property says that if a coalition is able to force the outcome of an interaction to lie in a particular set, then no possible combinations of moves by the other agents can prevent this to happen. We think that regularity is a key property to understand the meaning of ability. If an agent is properly able to do something this means that others have no means to prevent it. Outcome monotonicity is a property of all Effectivity Functions in CL, which is therefore a monotonic modal logic. It says that if a coalition is able to force the outcome of the interaction to lie in a particular set, then is also able to force the outcome to lie in all his supersets (see [6]). The last condition is IOEC, that assigns to the empty coalition the coarsest possible ability. As noticed also in [2] with such a property the empty coalition cannot force non-trivial outcomes of a game.

One important class of Effectivity Functions are the *playable* ones, introduced first in [6], to which we will refer throughout the paper.

Definition 3 (Playability)
For any world w an Effectivity Function is playable *if it has the following properties:*

1. *$\emptyset \notin E(w)(C)$, for any (C);*
2. *$W \in E(w)(C)$ for any C.*

3. *E is Agt-maximal, that is for any* $X \subseteq W$, *s.t.* $W \setminus X \notin E(w)(\emptyset)$ *implies* $X \in E(w)(Agt)$;
4. *E is superadditive, i.e. for* $C \cap D = \emptyset$, *if* $X \in E(w)(C)$ *and* $Y \in E(w)(D)$ *then* $X \cap Y \in E(w)(C \cup D)$.

The first condition imposes that interactions are nonempty, the second that coalitions can always choose the largest possible set, the third that the Grand Coalition of agents can do whatever not blocked by Nature, the fourth that coalitions can join their forces. Henceforth we will call Agt-*superadditive* those Effectivity Functions for which superadditivity holds for $C = \emptyset, D = Agt$, and we will call *almost-superadditive* those for which superadditivity holds for all coalitions C, D with $C \cup D \neq Agt$.

As proved in [6] [Theorem 2.27], nonempty strategic games exactly correspond to playable Effectivity Functions [1].

Playability and Coherence. What kind of interactions are coherent Effectivity Functions isolating?

In this respect it is interesting to compare playable and coherent Effectivity Function. For the proof of the propositions see the Appendix.

Proposition 1. *Not all playable EF are coherent, and not all coherent EF are playable.*

Proposition 2. *Coherent* Agt*-maximal EF are* Agt*-superadditive.*

Proposition 3. *Coherent Agt-maximal almost-superadditive EF are playable. Playable EF with IOEC are coherent.*

3 Coherent Coalition Logic and Its Axiomatization

In order to fully understand what sort of interactions we are investigating by using coherent Effectivity Functions we need to provide an axiomatization of their logic.

We now introduce the language and the models of Coherent Coalition Logic, that extends Coalition Logic with an auxiliary existential modality.

3.1 Language

Let Agt be a finite set of agents and $Prop$ a countable set of atomic formulas. The syntax of CCL is defined as follows:

$$\phi ::= p|\neg\phi|\phi \vee \phi|[C]\phi|E\phi$$

where p ranges over $Prop$ and C ranges over the subsets of Agt. The other boolean connectives are defined as usual. The informal reading of the modalities is: "Coalition C can choose ϕ" and "There is a state that satisfies ϕ". We abbreviate $\neg E\neg\phi$ with $A\phi$.

[1] The proof involves the definition of strategic game as a tuple $\langle N, \{\Sigma_i | i \in N\}, o, S\rangle$ where N is a set of players, each i being endowed with a set of strategies σ_i from Σ_i, an outcome function that returns the result of playing individual strategies at each of the states in S; the definition of α-Effectivity Function for a nonempty strategic game G, $E^\alpha_G : \wp(N) \to \wp\wp(S)$ defined as follows: $X \in E^\alpha_G$ iff $\exists\sigma_C \forall\sigma_{\overline{C}} o(\sigma_C; \sigma_{\overline{C}}) \in X$. The above mentioned theorem establishes that $E^\alpha_G = E$ in case E is playable and G is a nonempty strategic game.

3.2 Structures

Definition 4 (Models). *A* model *for our logic is a tuple*

$$(W, E, R_{\exists}, V)$$

where:

- W *is a nonempty set of states;*
- $E : W \longrightarrow (2^{Agt} \longrightarrow 2^{2^W})$ *is a coherent Effectivity Function.*
- $R_{\exists} = W \times W$ *is a global relation.*
- $V : W \longrightarrow 2^{Prop}$ *is a valuation function.*

3.3 Semantics

The satisfaction relation of modal formulas (the rest is standard) with respect to a pointed model M, w is defined as follows:

$$M, w \models [C]\phi \text{ iff } wE_C[[\phi]]^M$$
$$M, w \models E\phi \text{ iff } \exists v \text{ s.t. } wR_{\exists}v \text{ and } M, v \models \phi$$

$[[\phi]]^M =_{def} \{w \in W \mid M, w \models \phi\}$ is called the *truth set* or the *extension* of ϕ in M. The modality for coalitional ability is standard from CL [6].

4 On the Axiomatization of CCL

We recall the definition of Coalitional Canonical Model given in [6]. It is a model $M^* = ((W^*, E^*), V^*)$, where W^* is the set of all maximally consistent sets, that are closed under Modus Ponens and Monotonicity ($\phi \rightarrow \psi \Rightarrow [C]\phi \rightarrow [C]\psi$); E^* is the canonical relation and V^* the canonical valuation function. We denote with $\widehat{\phi} = \{w \in W^* | \phi \in w\}$ the truth set of ϕ in the canonical model. The canonical relation (the rest is standard) is defined as

$$wE^*_C X \text{ iff } \exists \phi : \widehat{\phi} \subseteq X \text{ and } [C]\phi \in w$$

The relation defined by Pauly is easily proved to be monotonic. Moreover the following theorem [[6] 3.10] holds: Every Coalition Logic Λ is sound and complete with respect to its canonical model M^*.

What we look for now is a set of axioms and rules such that the corresponding maximally consistent sets generate a coherent Effectivity Function in the canonical models.

While regularity and coalition monotonicity look quite straightforward to axiomatize, it is not so for inability of the empty coalition. It is rather clear that $[\emptyset]\phi \rightarrow [\emptyset](\phi \vee \psi)$ or also $[\emptyset]\top$ would not be appropriate axioms for CCL: they would both ensure the presence of the unit in the neighbourhood of $\emptyset$, but they would not say anything about the absence of all the other sets.

In fact the IOEC is not definable in CL. To see this it is important to notice that CL is monotonic multimodal logic, and frame validity of formulas of monotonic modal logics is closed under taking disjoint unions. This is proved for modal satisfaction in [4][Definition 4.1, Proposition 4.2]. For a definition see the Appendix (5).

Proposition 4. *There is no formula of CL that defines IOEC.*

As noticed before, IOEC intuitively describes a global property of the models, by requiring an accessibility relation that covers the whole domain.

We will give to this intuition a formal characterization, stating that in fact the ability of the empty coalition in CCL *is* a global modality.

4.1 IOEC Is a Global Relation

We claim that $IOEC$ is definable in CL plus the global modality.

Proposition 5. $A\phi \leftrightarrow [\emptyset]\phi$ *defines the IOEC. That is,*
$\models_C A\phi \leftrightarrow [\emptyset]\phi \Leftrightarrow E(w)(\emptyset) = \{W\}$ *for every* w *in the class of coalitional frames* C.

4.2 Axiomatization for the Global Modality Plus a New Inclusion Axiom

The global relation induces an equivalence class in the models, therefore it is axiomatizable by an $S5$ modality interpreted on a global relation. However this does not ensure that the underlying relation $R_\exists$ is globally connected. Global connectedness is not definable in basic modal language [1] [2].

As suggested in [1][p.417-418], given a set of maximally consistent formulae Σ^+ we can simply take a generated submodel of the canonical model in such a way that the truth of formulae in Σ^+ is preserved and the relation is (it follows by construction) a global relation. The definitions (6 for Basic Modal Language and 7 for Monotonic Modal Language) are reported in the Appendix.

Taken the canonical model $M^* = ((W^*, E^*, R^*_\exists), V^*)$, its submodel $M^{*'} = ((W^{*'}, E^{*'}, R^{*'}_\exists), V^*)$ generated by Σ^+ using the $R^*_\exists$ relation ensures that $R^{*'}_\exists = W^{*'} \times W^{*'}$.

Nevertheless in taking the generated submodel we should also ensure that the coalitional relation is not altered. One way to do so is to guarantee that the canonical coalitional relation is included in the global relation and that the generated submodel for the second relation is also a generated submodel for the first.

Modal satisfaction is invariant under taking generated submodels both in the Basic Modal and in the Monotonic Modal case, that is, for all states of the generated submodels, truth of modal formulas is preserved [1] [4]. It has to be noticed though that the constructions are different, and it is by no means automatic that truth of a formula preserved in one case transfers to the other.

Therefore, in our case, the question is whether the submodel generated using the existential modality through a maximally consistent set of formulas Σ^+ (making the canonical model strongly connected with respect to this relation) is also a generated submodel with respect to the coalitional relation.

[2] The reason is also the invariance under taking disjoint unions. This fact sheds light on the relation between IOEC and Global Relation, in fact now we see clearly that the ability of the empty coalition in CCL *is* a global modality.

The answer is: it depends on the extra axioms. Usually when we have a K modality and a global modality it is sufficient to include the normal relation corresponding to K in the global relation corresponding to the global modality. But we cannot simply have:

$$[C]\phi \rightarrow E\phi$$

because the canonical relation for the neighbourhood modality may cross $S5$ equivalence classes. Instead the good candidate for our attempt is just the following:

$$A\phi \leftrightarrow [\emptyset]\phi$$

We claim that taking a generated submodel with respect to the global relation, given this axiom, ensures the condition of taking also a generated submodel with respect to the neighbourhood modality.

Proposition 6. *The axiom $A\phi \leftrightarrow [\emptyset]\phi$ guarantees inclusion of the canonical relation in the global relation.*

Now let us take a generated submodel, as described in [1] for basic modal logic, using the maximally consistent set Σ^+ looking only at the global modality.

Proposition 7. *The generated canonical submodel under Σ^+ preserves both global modality and monotonic CL formulas satisfaction.*

It follows that we have an axiomatization for CCL.

4.3 A Sound and Complete Axiomatization

Take now the maximally consistent sets $w \in W^*$, closed under the proof system depicted in the table.

We take the following conditions to describe coherence of the Effectivity Function on the canonical relation.

- $wE^*_C X$ iff $\exists \widehat{\phi} \subseteq X : [C]\phi \in w$ and $\forall \widehat{\psi} \subseteq (W^* \setminus X) : [\overline{C}]\psi \notin w$ (for $C \neq \emptyset$)
- $E^*_C \subseteq E^*_D$ (for $C \subseteq D$)
- $wE^*_C X$ iff $X = W^*$ (for $C = \emptyset$)
- $wR_\exists v$ iff $w, v \in W^*$

Notice that the following proposition follows from what shown before:

Proposition 8. *The canonical coherent frame for CL with $A\phi \leftrightarrow [\emptyset]\phi$ as axiom has the property that $E(w)(\emptyset) = W^*$ for any MCS w and $A\phi \leftrightarrow [\emptyset]\phi$ is valid in the class of frames with that property.*

We are now ready to prove soundness and completeness.

Proposition 9. *The set of axioms and rules in Table 3 are sound and complete with respect to Coherent Coalition Frames.*

Table 3. Proof System

A1	$[C]\phi \rightarrow [D]\phi$ (for $C \subseteq D$)
A2	$[C]\phi \rightarrow \neg[\overline{C}]\neg\phi$
A3	$A\phi \leftrightarrow [\emptyset]\phi$
A4	$\phi \rightarrow E\phi$
A5	$EE\phi \rightarrow E\phi$
A6	$\phi \rightarrow AE\phi$
A7	$A(\phi \rightarrow \psi) \rightarrow (A\phi \rightarrow A\psi)$
R1	$\phi \wedge (\phi \rightarrow \psi) \Rightarrow \psi$
R2	$\phi \rightarrow \psi \Rightarrow [C]\phi \rightarrow [C]\psi$
R3	$\phi \Rightarrow A\phi$

4.4 On Agt-Maximal Coherent EF

If we add Agt-maximality to Coherent EF, the following holds:

$$M, w \models [Agt]\phi \leftrightarrow E\phi$$

This suggest that the augmented version of CCL is powerful enough to reason on global properties of the models. Moreover a new axiomatization is easy to be obtained.

Proposition 10. *Agt-Maximal CCL is sound and complete with respect to the axiom system of CCL plus* $[Agt]\phi \leftrightarrow E\phi$ *and, as for the canonical relation, the condition* $wE^*_C X$ iff $\exists\widehat{\phi}$ s.t. $\widehat{\phi} \subseteq X \subseteq W^*$ and $[C]\phi \in w$ (for $C = Agt$) *is added.*

These results are useful to apply the language to the study of Multi-Agent interactions. Therefore many examples from Game Theory, such as Coordination Game or Prisoner Dilemma, are instances of coherent interactions.

5 Discussion

Deontic CCL Once we view a deontic language as regulating a Multi Agent System, we can say that a set of commands promote a certain interaction (or social state), prohibiting certain others. Following this line of reasoning it is possible, given a notion of optimality or efficiency, to construct a deontic language that requires this notion to hold.

Suppose we had a deontic operator O, such that $O(C, \phi)$ would hold in a model iff the formula ϕ were a choice of coalition C leading to an efficient outcome[3].

If we take a model M_c of the game depicted in table 1, we would have that

$$M_c \models O(\emptyset, \phi) \leftrightarrow A\phi$$

while

$$M_c \models O(Agt, (white_{Agt}) \vee (black_{Agt}))$$

[3] We take this view in [3]

where the propositions have the obvious reading. Thus the regulation would impose to Nature just what is already valid in the model, whereas it would tell the agents how to behave to reach an efficient state. Extending the example further on, we can think of a deontic operator F such that $F(C, \phi)$ would hold in a model iff all choices of C necessarily leading to an inefficient outcome were ϕ choices. Assuming that the System is indeed drivable to an efficient outcome (there is one) we would have that

$$M_c \models F(\emptyset, \bot)$$

while

$$M_c \models F(Agt, (white_{Agt}) \wedge (black_{Agt}))$$

In this case and, provided the existence of an efficient outcome, in general, Nature is never forbidden anything, while prohibitions of Deontic CCL always address nonempty coalitions of agents.

6 Conclusion and Future Work

In this paper we studied those interactions in which Nature does not play an active role, and we provided an axiomatization of the resulting logic, discussing the possible applications in game-theoretical scenarios. The work allows for several developments.

The most straightforward seems the switch from game forms (interactions without specified preferences for players) to games, by specifying a preference relation. The interaction between these two relations could shed light on the understanding of the various shapes of motivational attitudes in interaction (see in this respect [7]). In this respect, one more issue concerns the relation between coalition logics, interpreted in neighbourhood models, and normal modalities, used for instance to talk about preferences or as we did here to talk about global properties of the models. In our paper we show that the logic of certain effectivity function simulate the universal modality. It would be interesting to bring this correspondence further, as suggested for instance in [4]. The study of interactions was shown to have an interesting connection with deontic logic that, viewed in a Multi-Agent perspective, allows to talk about those desirable properties that an interaction should have. As system designers, our aim is at last to construct efficient social procedures that can guarantee a socially desirable property to be reached. We think that normative system design is at last a proper part of the Social Software enterprise [5].

References

1. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science (2001)
2. Borgo, S.: Coalitions in action logic. In: Proc. of IJCAI, pp. 1822–1827 (2007)
3. Broersen, J., Mastop, R., Meyer, J.-J.Ch., Turrini, P.: A deontic logic for socially optimal norms (forthcoming, 2008)
4. Hansen, H.H.: Monotonic Modal Logics. Master Thesis, ILLC (2001)
5. Parikh, R.: Social software. Synthese 132(3), 187–211 (2002)
6. Pauly, M.: Logic for Social Software. ILLC Dissertation Series (2001)
7. von Wright, G.H.: The logic of preference. Edimburgh University Press (1963)

A Appendix I: Definitions

Definition 5 (Disjoint Unions for MML)
Let $M_i = (W_i, N_i, V_i), i \in I$, *be a collection of disjoint models. Then we define their* disjoint union *as the model* $\oplus M_i = (W, N, V)$ *where* $W = \bigcup_{i\in I} W_i, V(p) = \bigcup_{i\in I} V_i(p)$ *and for* $X \subseteq W, w \in W_i$,

$$X \in N(w) \textit{ iff } X \cap W_i \in N_i(w)$$

Definition 6 (Generated Submodels for BML)
Let $M = (W, R, V)$ *and* $M' = (W', R', V')$ *be two models; we say that* M' *is a submodel of* M *if* $W \subseteq W'$, R' *is the restriction of* R *to* W', *that is* $R' = R \cap (W' \times W')$ *and* V' *is the restriction of* V *to* M'. *We say that* M' *is a generated submodel of* M $(M' \mapsto M)$ *if* M' *is a submodel of* M *and for all worlds* w, v *the following closure condition holds:*

$$\textit{if } w \textit{ is in } M' \textit{ and } Rwv, \textit{ then } v \textit{ is in } M'$$

Definition 7 (Generated Submodels for MML). *Given a monotonic model* M, M' *is a submodel of* M *if* $W' \subseteq W, V'(p) = V(p) \cap W'$ *for* p *atomic, and* $N' = N \cap (W' \times 2^{W'})$, *that is*

$$\forall s \in W' : N'(s) = \{X \subseteq W' | X \in N(s)\}$$

In neighbourhood semantics given M' *submodel of* M, M' *is also a generated submodel of* M *if the identity mapping* $i : W \to W'$ *is a bounded morphism, that is, for all* $w' \in W'$ *and all* $X \subseteq W$

$$i^{-1}[X] = X \cap W' \in N'(w') \textit{ iff } X \in N(w')$$

B Appendix II: Selected Proofs

Proof of Proposition 1. For the first part, take $W = \{x, y\}, Agt = \{i, j\}$ and the Effectivity Function $E(k)(\emptyset) = E(k)(\{i\}) = E(k)(\{j\}) = E(k)(Agt) = \{W, W \setminus \{x\}\}$ for $k \in W$. Now it is just a matter of checking the conditions for playability.

For the second part take $W = \{x, y\}, Agt = \{i, j\}$ with $E(k)(\emptyset) = E(k)(\{i\}) = E(k)(\{j\}) = E(k)(Agt) = \{W\}$ for $k \in W$.

Proof of Proposition 2. Suppose $A \in E(w)(\emptyset)$ and $B \in E(w)(Agt)$ for an arbitrary w. We prove the claim by reductio. So assume $A \cap B \notin E(w)(Agt)$. By Agt-maximality $W \setminus (A \cap B) = W' \in E(w)(\emptyset)$. But by regularity it is not possible that $A \cap B = \emptyset$. Then $W' \subset W$, contradicting inability of the empty coalition.

Proof of Proposition 3. The first part follows from the previous proposition and from the definition of playability. The second from the definition of coherence and the properties of playability.

Proof of Proposition 4. Without loss of generality, we can simply think of the monotonic modal logic with only the box for the empty coalition, and take frames instead of models.

Consider the following monotonic frames $F_0 = (W_0, N_0)$ and $F_1 = (W_1, N_1)$, with a domain W_j and a relation $N_j \subseteq W_j \times 2^{W_j} (j \in \{0, 1\})$. Take $W_0 = \{w_0\}$, $W_1 = \{w_1\}$, $N_0(w_0) = \{w_0\}$ and $N_1(w_1) = \{w_1\}$. Now suppose ϕ is some formula true at a world w in a model $M' = (W', N', V')$ of a monotonic frame F' iff $[[\top]]^M$ is neighbour of w (if $wN'[[\top]]$) and nothing else is ($wN'X \Rightarrow X = [[\top]]$). We see that $M_0, w_0 \models \phi$ and $M_1, w_1 \models \phi$ for arbitrary M_i inside F_i ($i \in \{0, 1\}$). From [4] we construct the disjoint union $\oplus(F_0, F_1) = (W, N)$ as defined in the Appendix. We see clearly that our formula ϕ is not true in the disjoint union, because the neighbourhoods of the single models are copied in the disjoint union even if they are smaller than the unit. We observe moreover that the disjoint union is monotonic. The conclusion is that the formula expressing inability of the empty coalition is not definable is monotonic modal language.

Proof of Proposition 5. ($\Rightarrow$) Assume that $\models_C A\phi \leftrightarrow [\emptyset]\phi$ while not $E(w)(\emptyset) = \{W\}$ for every w in any frame F in the class of Coalitional Frames C. Then there is an F in which there is a w such that $E(w)(\emptyset) \neq \{W\}$. Notice that both W and $E(w)(\emptyset)$ are nonempty. So there is a $W' \neq W$ s.t $W' \in E(w)(\emptyset)$. Take an atom p to be false in all $w' \in W'$ and true in $W \setminus W'$. Now we have model M based on a coalitional frame C for which $M \not\models Ap \leftrightarrow [\emptyset]p$. Contradiction.

($\Leftarrow$) Assume $E(w)(\emptyset) = \{W\}$ for a given w in an arbitrary model M of a coalition frame in C, and that $w \models A\phi$. Then $[[\phi]]^M = W$ and $w \models [\emptyset]\phi$ follows. Assume now that $w \models [\emptyset]\phi$. It has to be the case that $[[\phi]]^M = W$ by assumption. So also that $w \models A\phi$, which concludes the proof.

Proof of Proposition 6. Take a maximally consistent set of formulas Σ^+ in a canonical model $M^* = ((W^*, E^*, R^*_\exists), V^*)$ that extends a consistent set of formulas Σ according to the axioms and the rules that we have defined for the global and the standard coalitional modality plus $A\phi \leftrightarrow [\emptyset]\phi$. Suppose now $A\phi$ is in Σ^+ for some ϕ. This means that $W^* = [[\phi]]^{M^*}$. Now take some $[C]\psi$ in the same maximally consistent set of formulas. This means that $[[\psi]]^{M^*} \in E^*(\Sigma^+)(C)$. But by definition, $[[\psi]]^{M^*} \subseteq W^*$ which proves that all coalitional neighbourhoods are covered by the global modality relation.

Proof of Proposition 7. It is just a matter of verifying that the generated submodel for the global relation is also a generated submodel for the coalitional relation. We in fact did not change any neighbourhood relation, that are just copied into the generated submodel.

Proof of Proposition 9. We need just to check the statement with respect to $M^{*'}$. We omit the detailed proof.

Proof of Proposition 10. The argument is a straightforward adaptation of the previous proofs.

Declarative Semantics for Revision Programming and Connections to Active Integrity Constraints

Luciano Caroprese[1] and Mirosław Truszczyński[2]

[1] Università della Calabria, 87030 Rende, Italy
caroprese@deis.unical.it
[2] Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA
mirek@cs.uky.edu

Abstract. We investigate *revision programming*, a formalism to describe constraints on belief sets (databases, knowledge bases), and to specify *preferred* ways to enforce them. We propose several semantics for revision programs combining ideas from logic programming and *active integrity constraints*, a formalism to model preferred ways to enforce integrity constraints on databases. We present results on the complexity of the semantics we introduce. We also show that all these semantics are invariant under "shifting". Finally, we prove that from the perspective of a broad semantic landscape of revision programming, there is a direct correspondence between revision programs and active integrity constraints.

1 Introduction

Revision programming [1,2] was proposed as a formalism to describe revisions of sets (state descriptions, belief sets, databases, etc). To focus our attention, we will discuss revision programs here in the context of databases but, as just noted, their applicability goes beyond that setting.

Revision programs consist of *revision rules*. They specify conditions a revised database has to satisfy, as well as preferred ways to enforce them. To describe formally the meaning of normal revision programs, researchers proposed the semantics of *justified* and *supported* revisions [2], motivated by the stable-model semantics [3] and the supported-model semantics [4] of logic programs, respectively. The semantics of justified revisions was later generalized to the case of disjunctive revision programs in [5]. In this paper, for consistency with other notation, we refer to it as the semantics of justified *weak* revisions.

The definitions of the two semantics of revision programs do not directly take into account the postulate of the *minimality of change*, which prefers as the results of revision those databases that differ minimally from the original ones. Not surprisingly, the postulate holds neither for justified weak revisions, a shortcoming that has apparently been overlooked so far, nor for the semantics of supported revisions. Second, while in the restricted case of *normal* revision programs justified weak revisions are change-minimal, it is a side-effect of other aspects of the definition rather than a result of an explicit design decision. Since the minimality of change is among the most basic principles

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 100–112, 2008.

of database update, belief revision and nonmonotonic reasoning (cf. for instance, [6,7]), it is important to study its effect on the semantics of revision programs. This is one of the objectives of this paper.

The theory of revision programs we develop here follows our recent investigations of *active integrity constraints* [8]. An *integrity constraint* is a condition on a database. If the database violates an integrity constraint, it needs to be *repaired* — updated so that the integrity constraint holds again. Often, there are several ways to do so. An *active* integrity constraint encodes *explicitly* both an integrity constraint and preferred basic actions to repair it, if it is violated. To specify the meaning of active integrity constraints, [9] proposed the notion of a *founded repair*. Founded repairs are change-minimal and satisfy a certain groundedness condition. In [8], we proposed several additional semantics for active integrity constraints. Together with founded repairs, they cover a spectrum of features one might require of database repairs. The class consists of the semantics of weak repairs, repairs, founded weak repairs, founded repairs, justified weak repairs, and justified repairs. The term *weak* points to the fact that the corresponding semantics is not required to have the minimality of change property. The terms *founded* and *justified* refer to different "grounding" principles used in defining the semantics.

We show[1] that this general schema for defining repairs can be lifted to revision programs, and yields the semantics of weak revisions, revisions, founded weak revisions, founded revisions, justified weak revisions, and justified revisions. We show that the semantics of founded weak revisions is an extension of the semantics of supported revisions to the case of disjunctive revision programs, and observe that the semantics presented in [5] appears in the schema under the name of justified *weak* revisions, as it does not satisfy the minimality-of-change property.

The relationship to active integrity constraints is not coincidental. Active integrity constraints, while different in several technical details, share with revision programs the same basic motivation of guiding the database update process through user-specified preferences. To make the connection between the two formalisms precise and explicit, we present a mapping from active integrity constraints to revision programs, under which the corresponding semantics on each side coincide.

The class of revision programs is in a certain sense broader than the class of active integrity constraints. There is a precise match between the two, however, if we limit attention to a subclass of revision programs that we refer to as *proper*. We prove that the restriction does not affect the expressive power of revision programming. Thus, our results show that both formalisms, even though syntactically different and originally endowed with different semantics, can be regarded as notational variants of each other.

One of the properties we establish for all semantics we discuss here is *invariance under shifting*. We use it to relate revision programs and logic programs of Lifschitz and Woo [10], aligning justified weak repairs with answer sets. We also point out that all other semantics of revision programs can be adapted for Lifschitz-Woo programs.

[1] We omit proofs due to space restrictions. They can be found at www.ca.uky.edu/ai/rp-full.pdf.

2 Revision Programming — An Overview

In this section we review the basic terminology of revision programming, and recall the two semantics introduced in [1,2,5]: the semantics of supported revisions, and the semantics of justified weak revisions (originally referred to in [5] as justified revisions and renamed here for consistency with the general naming schema we use).

We consider a finite set At of propositional atoms, representing all *ground* atoms in a language of predicate logic with a finite set of constants and with no function symbols. *Databases* are subsets of At. A *revision literal* is an expression $\mathbf{in}(a)$ or $\mathbf{out}(a)$, where $a \in At$. Revision literals $\mathbf{in}(a)$ and $\mathbf{out}(a)$ are *duals* of each other. If α is a revision literal, we denote its dual by α^D. We extend this notation to sets of revision literals. We say that a set of revision literals is *consistent* if it does not contain a pair of dual literals. Revision literals represent elementary updates one can apply to a database. We define the result of applying a *consistent* set $\mathcal{U}$ of revision literals to a database $\mathcal{I}$ as follows:

$$\mathcal{I} \oplus \mathcal{U} = (\mathcal{I} \cup \{a \mid \mathbf{in}(a) \in \mathcal{U}\}) \setminus \{a \mid \mathbf{out}(a) \in \mathcal{U}\}.$$

A *revision rule* is an expression of the form

$$r = \alpha_1 | \ldots | \alpha_k \leftarrow \beta_1, \ldots, \beta_m, \tag{1}$$

where $k, m \geq 0$, $k+m \geq 1$, and α_i and β_j are revision literals. The set $\{\alpha_1, \ldots, \alpha_k\}$ is the *head* of the rule (1); we denote it by $head(r)$. Similarly, the set $\{\beta_1, \ldots, \beta_m\}$ is the *body* of the rule (1); we denote it by $body(r)$. A revision rule is *normal* if $|head(r)| \leq 1$. A *revision program* is a collection of revision rules. A revision program is *normal* if all its rules are normal.

A database $\mathcal{D}$ *satisfies* a revision literal $\mathbf{in}(a)$ ($\mathbf{out}(b)$, respectively), if $a \in \mathcal{D}$ ($b \notin \mathcal{D}$, respectively). A database $\mathcal{D}$ *satisfies* a revision rule (1) if it satisfies at least one literal α_i, $1 \leq i \leq k$, whenever it satisfies every literal β_j, $1 \leq j \leq m$. Finally, a database $\mathcal{D}$ satisfies a revision program P, if $\mathcal{D}$ satisfies every rule in P. We use the symbol $\models$ to denote the satisfaction relation.

For revision literals $\alpha = \mathbf{in}(a)$ and $\beta = \mathbf{out}(b)$, we set $lit(\alpha) = a$ and $lit(\beta) = not\, b$. We extend this notation to sets of revision literals. We note that every database interprets revision literals and the corresponding propositional literals in the same way. That is, for every database $\mathcal{I}$ and for every set of revision literals L, $\mathcal{I} \models L$ if and only if $\mathcal{I} \models lit(L)$.

It follows that a revision rule (1) specifies an integrity constraint equivalent to the propositional formula: $lit(\beta_1), \ldots, lit(\beta_m) \supset lit(\alpha_1), \ldots, lit(\alpha_k)$. However, a revision rule is not only an integrity constraint. Through its syntax, it also encodes a preference on how to "fix" a database, when it violates the constraint. Not satisfying a revision rule r means satisfying all revision literals in the body of r *and* not satisfying any of the revision literals in the head of r. Thus, enforcing the constraint means constructing a database that (1) does not satisfy some revision literal in the body of r, *or* (2) satisfies at least one revision literal in the head of r. The underlying idea of revision programming is to prefer those revisions that result in databases with the property (2).

As an example, let us consider the revision rule $r = \mathbf{in}(a) \leftarrow \mathbf{out}(b)$, and the empty database $\mathcal{I}$. Clearly, $\mathcal{I}$ does not satisfy r. Although $\mathcal{I}$ can be fixed either by inserting

a, so that $head(r)$ becomes *true*, or by inserting b, so that $body(r)$ becomes *false*, the syntax of r makes the former preferred.

Normal revision programs were introduced and studied in [1,2] and the semantics of supported and justified weak revisions for normal revision programs were proposed there. In [5], the formalism was extended to allow disjunctions of revision literals in the heads of rules, and the semantics of justified weak revisions was generalized to that case. We will now recall these definitions.

First, we define the notion of the *inertia set*. Let $\mathcal{I}$ and $\mathcal{R}$ be databases. We define the *inertia set* wrt $\mathcal{I}$ and $\mathcal{R}$, denoted $I(\mathcal{I},\mathcal{R})$, by setting

$$I(\mathcal{I},\mathcal{R}) = \{\mathbf{in}(a) \mid a \in \mathcal{I} \cap \mathcal{R}\} \cup \{\mathbf{out}(a) \mid a \notin \mathcal{I} \cup \mathcal{R}\}.$$

In other words, $I(\mathcal{I},\mathcal{R})$ is the set of all *no-effect* revision literals for $\mathcal{I}$ and $\mathcal{R}$, that is, revision literals that have no effect when revising $\mathcal{I}$ into $\mathcal{R}$.

Now, let P be a *normal* revision program and $\mathcal{R}$ be a database. By $P_{\mathcal{R}}$ we denote the program obtained from P by removing each rule $r \in P$ such that $\mathcal{R} \not\models body(r)$.

Definition 1. [SUPPORTED UPDATES AND SUPPORTED REVISIONS]. *Let P be a normal revision program and $\mathcal{I}$ a database. A set $\mathcal{U}$ of revision literals is a* supported update *of $\mathcal{I}$ wrt P if $\mathcal{U}$ is consistent and $\mathcal{U} = head(P_{\mathcal{I}\oplus\mathcal{U}})$. A set $\mathcal{E}$ is a* supported revision *of $\mathcal{I}$ wrt P if $\mathcal{E} = \mathcal{U} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$.* □

Intuitively, a consistent set $\mathcal{U}$ of revision literals is a supported update if it is precisely the set of literals "supported" by P and the database resulting from updating $\mathcal{I}$ with $\mathcal{U}$. Eliminating from a supported revision all no-effect literals yields a supported revision.

While not evident explicitly from the definition, supported updates and revisions guarantee constraint enforcement (cf.[2]).

Proposition 1. *Let P be a normal revision program and $\mathcal{I}$ a database. If $\mathcal{E}$ is a supported revision of P, then $\mathcal{I} \oplus \mathcal{E} \models P$.* □

Supported updates do not take into account the inertia set. Supported revisions do, but only superficially: simply removing no-effect literals from the corresponding supported update. Consequently, supported updates and revisions allow for a possibility for circularity of support and non-minimality.

Example 1. Let P be a revision program containing the rules $\{\mathbf{in}(a) \leftarrow \mathbf{in}(b),\ \mathbf{in}(b) \leftarrow \mathbf{in}(a), \mathbf{in}(c) \leftarrow \mathbf{out}(d)\}$, and let $\mathcal{I}$ the empty database. $\mathcal{I}$ does not satisfy P as it violates the rule $\mathbf{in}(c) \leftarrow \mathbf{out}(d)$. One can check that set $\mathcal{U} = \{\mathbf{in}(a), \mathbf{in}(b), \mathbf{in}(c)\}$ modeling the insertions of a, b and c, is a supported update and a supported revision. However it is not minimal as its subset $\{\mathbf{in}(c)\}$ is sufficient to guarantee the satisfaction of P. □

The problem in the previous example is the circularity of support between $\mathbf{in}(a)$ and $\mathbf{in}(b)$. Each of them supports the other one but the set containing both is superfluous. To address the problem, [1,2] proposed for normal revision programs the semantics of justified weak revisions, later extended to the disjunctive case in [5]. The idea was to "ground" justified weak revisions in the program and the inertia set by means of a *minimal closure*.

A set $\mathcal{U}$ of revision literals is *closed* under a revision program P (not necessarily normal) if for every rule $r \in P$, whenever $body(r) \subseteq \mathcal{U}$, then $head(r) \cap \mathcal{U} \neq \emptyset$. If $\mathcal{U}$ is closed under P and for every set $\mathcal{U}' \subseteq \mathcal{U}$ closed under P, we have $\mathcal{U}' = \mathcal{U}$, then $\mathcal{U}$ is a *minimal closed* set for P. Clearly, in general a revision program may admit more than one minimal closed set of revision literals.

Definition 2. [JUSTIFIED UPDATES AND JUSTIFIED WEAK REVISIONS]. *Let P be a revision program and let $\mathcal{I}$ be a database. A consistent set $\mathcal{U}$ of revision literals is a* P-justified update *for $\mathcal{I}$ if it is a minimal set closed under $P \cup I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$.*

If $\mathcal{U}$ is a P-justified update for $\mathcal{I}$, then $\mathcal{U} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$ is a P-justified weak revision *for $\mathcal{I}$.* □

The inertia set plays an essential role in the definition, as it is used directly in the definition of a P-justified update. Again, it is not self-evident from the definition that justified updates and justified weak revisions, when applied to an initial database yield a database satisfying the program. However, the definition does indeed imply so [2,5].

Proposition 2. *Let P be a revision program and $\mathcal{I}$ a database. If $\mathcal{U}$ is a justified update or justified weak revision of P, then $\mathcal{I} \oplus \mathcal{U} \models P$.* □

3 A Family of Declarative Semantics for Revision Programming

The two semantics in the previous section were defined based on how revisions are "grounded" in a program, an initial database, and the inertia set. Two fundamental postulates of constraint enforcement and minimality of change played no explicit role in that research. The first one is no problem as it is a side effect of each of the two types of groundedness considered (cf. Propositions 1 and 2). The second one simply does not hold for supported revisions. And while [2] proved that justified weak revisions are change-minimal in the case of *normal* revision programs, it is not so in the general case.

Example 2. Let P be a revision program consisting of the rules $\{\mathbf{in}(a)|\mathbf{out}(b) \leftarrow \mathbf{out}(c), \mathbf{out}(a)|\mathbf{in}(b) \leftarrow \mathbf{out}(c)\}$ and the $\mathcal{I}$ the empty database. It is easy to verify that set $\mathcal{U} = \{\mathbf{in}(a), \mathbf{in}(b)\}$ is a justified weak revision. However it is not minimal as $\mathcal{I}$ is already consistent and no update is needed. □

We will now develop a range of semantics for revision programs by taking the postulates of constraint enforcement and minimality of change explicitly into consideration.

Definition 3. [WEAK REVISIONS AND REVISIONS]. *A set $\mathcal{U}$ of revision literals is a* weak revision *of $\mathcal{I}$ wrt a revision program P if (1) $\mathcal{U} \cap I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U}) = \emptyset$ (relevance — all revision literals in $\mathcal{U}$ actually change $\mathcal{I}$ or, in other words, none of them is a no-effect literal wrt $\mathcal{I}$ and $\mathcal{I} \oplus \mathcal{U}$); and (2) $\mathcal{I} \oplus \mathcal{U} \models P$ (constraint enforcement). Further, $\mathcal{U}$ is a* revision *of $\mathcal{I}$ wrt a revision program P if it is a weak revision and for every $\mathcal{U}' \subseteq \mathcal{U}$, $\mathcal{I} \oplus \mathcal{U}' \models P$ implies that $\mathcal{U}' = \mathcal{U}$ (minimality of change).* □

Example 3. Let P be a revision program consisting of a rule $\mathbf{out}(b) \leftarrow \mathbf{in}(a)$, and let $\mathcal{I} = \{a, b\}$ be a database. Clearly, $\mathcal{I}$ does not satisfy P. The program P has three weak revisions: $\mathcal{U}_1 = \{\mathbf{out}(a)\}$, $\mathcal{U}_2 = \{\mathbf{out}(b)\}$ and $\mathcal{U}_3 = \{\mathbf{out}(a), \mathbf{out}(b)\}$, respectively. The sets $\mathcal{U}_1$ and $\mathcal{U}_2$ are revisions. The set $\mathcal{U}_3$ is not. □

(Weak) revisions do not reflect the preferences on how to revise a database encoded in the syntax of revision rules. We will now introduce semantics that aim to capture that preference. First, we define a new semantics for revision programs by imposing change-minimality on justified weak revisions.

Definition 4. [JUSTIFIED REVISIONS]. *Let P be a revision program and let $\mathcal{I}$ be a database. A P-justified weak revision $\mathcal{E}$ for $\mathcal{I}$ is a* P-justified revision *for $\mathcal{I}$ if $\mathcal{E}$ is a revision of $\mathcal{I}$ wrt P (that is, for every set $\mathcal{E}' \subseteq \mathcal{E}$ such that $\mathcal{I} \oplus \mathcal{E}' \models P$, $\mathcal{E}' = \mathcal{E}$).* □

Justified revisions have several useful properties. They are change-minimal and are grounded in the program *and* the inertia set. However, as stable models of logic programs, with which they share several similarities, justified revisions are not designed to handle reasoning by cases.

Example 4. Let $P = \{\mathbf{in}(b) \leftarrow \mathbf{in}(a),\ \mathbf{in}(b) \leftarrow \mathbf{out}(a),\ \mathbf{in}(a) \leftarrow \mathbf{in}(b)\}$ and let $\mathcal{I} = \emptyset$. A possible interpretation of the first two revision rules could be that no matter what the status of a is, b must be in the database. By the third rule a must belong to the database, too. Thus, the program justifies the set $\mathcal{R} = \{\mathbf{in}(a), \mathbf{in}(b)\}$ as a revision of $\mathcal{I}$ (assuming that we allow reasoning by cases). It is easy to verify that $\mathcal{R} = \{\mathbf{in}(a), \mathbf{in}(b)\}$ is a revision of $\mathcal{I}$. However, it is not P-justified (weak) revision of $\mathcal{I}$. □

To provide a semantics capturing such justifications, we introduce now the concept of foundedness and the semantics of founded (weak) revisions,

Definition 5. [FOUNDED (WEAK) REVISIONS]. *Let $\mathcal{I}$ be a database, P a revision program and, and $\mathcal{E}$ a consistent set of revision literals.*

1. *A revision literal α is* P-founded *wrt $\mathcal{I}$ and $\mathcal{E}$ if there is $r \in P$ such that $\alpha \in head(r)$, $\mathcal{I} \oplus \mathcal{E} \models body(r)$, and $\mathcal{I} \oplus \mathcal{E} \models \beta^D$, for every $\beta \in head(r) \setminus \{\alpha\}$.*
2. *The set $\mathcal{E}$ is* P-founded *wrt $\mathcal{I}$ if every element of $\mathcal{E}$ is P-founded wrt $\mathcal{I}$ and $\mathcal{E}$.*
3. *$\mathcal{E}$ is a* P-founded (weak) revision *for $\mathcal{I}$ if $\mathcal{E}$ is a (weak) revision of $\mathcal{I}$ wrt P and $\mathcal{E}$ is* P-founded *wrt $\mathcal{I}$.* □

One can verity that revision $\mathcal{R}$ in Example 4 is founded. We also note that condition (3) of the definition guarantees that founded (weak) revisions enforce constraints of the revision program. Next, directly from the definition, it follows that founded weak revisions are weak revisions. Similarly, founded revisions are revisions and so, they are change-minimal. Furthermore, founded revisions are founded weak revisions. However, there are (weak) revisions that are not founded, and founded weak revisions are not necessarily founded revisions, that is, are not change-minimal. The latter observation shows that foundedness is too weak a condition to guarantee change-minimality.

Example 5. Let P be the revision program containing the rules $\{\mathbf{in}(b) \leftarrow \mathbf{in}(a),\ \mathbf{in}(a) \leftarrow \mathbf{in}(b), \mathbf{in}(c) \leftarrow \mathbf{out}(d)\}$ and $\mathcal{I}$ the empty database. The set $\{\mathbf{in}(d)\}$ is a revision of $\mathcal{I}$ wrt P. Therefore it is a weak revision of $\mathcal{I}$ wrt P. However, it is not a P-founded weak revision for $\mathcal{I}$. Therefore, it is not a P-founded revision for $\mathcal{I}$, either. The set $\{\mathbf{in}(c), \mathbf{in}(a), \mathbf{in}(b)\}$ is a P-founded weak revision for $\mathcal{I}$ but not a P-founded revision for $\mathcal{I}$. Indeed, $\{\mathbf{in}(c)\}$ is also a revision of $\mathcal{I}$ wrt P. □

In the case of normal revision programs, founded weak revisions coincide with supported revisions.

Theorem 1. *Let P be a normal revision program and $\mathcal{I}$ a database. A set $\mathcal{E}$ of revision literals is a P-founded weak revision of $\mathcal{I}$ if and only if $\mathcal{E}$ is a P-supported revision of $\mathcal{I}$.* □

Foundedness is less restrictive than the condition defining justified updates, which is behind justified (weak) revisions.

Theorem 2. *Let P be a revision program and let $\mathcal{I}$ be a database. If a set $\mathcal{E}$ of revision literals is a P-justified (weak) revision of $\mathcal{I}$, then it is a P-founded (weak) revision of $\mathcal{I}$.* □

The converse implications do not hold in general (cf. Example 4).

To summarize our discussion so far, revision programs can be assigned the semantics of (weak) revisions, justified (weak) revisions and founded (weak) revisions. Let us denote the classes of the corresponding types of revisions by $\mathbf{Rev}(\mathcal{I}, P)$, $\mathbf{WRev}(\mathcal{I}, P)$, $\mathbf{JRev}(\mathcal{I}, P)$, $\mathbf{JWRev}(\mathcal{I}, P)$, $\mathbf{FRev}(\mathcal{I}, P)$ and $\mathbf{FWRev}(\mathcal{I}, P)$. The containment relations are demonstrated in Figure 1. None of the containment relations can be replaced with the equality.

$$\begin{array}{ccccc} \mathbf{JRev}(\mathcal{I},P) & \subseteq & \mathbf{FRev}(\mathcal{I},P) & \subseteq & \mathbf{Rev}(\mathcal{I},P) \\ \cap\!| & & \cap\!| & & \cap\!| \\ \mathbf{JWRev}(\mathcal{I},P) & \subseteq & \mathbf{FWRev}(\mathcal{I},P) & \subseteq & \mathbf{WRev}(\mathcal{I},P) \end{array}$$

Fig. 1. The containment relations for the semantics for revision programs

4 Properties of the Semantics

In this section, we will present several properties of the semantics we introduced here. Specifically, we exhibit two cases when justified weak revisions and justified revisions coincide, we present some complexity results and discuss the shifting property.

Programs whose justified weak revisions are change-minimal. In general, justified weak revisions are not change-minimal. However, for normal revision programs, justified weak revisions are change-minimal [2]. The change-minimality holds also in the following setting.

Theorem 3. *Let $\mathcal{I}$ be a database and P a revision program such that for each revision literal $\alpha \in \bigcup_{r \in P} head(r)$, $\mathcal{I} \models lit(\alpha^D)$. If $\mathcal{E}$ is a P-justified weak revision of $\mathcal{I}$, then $\mathcal{E}$ is a P-justified revision of $\mathcal{I}$.* □

Theorem 3 concerns the case when each revision literal in the head of a revision rule, if applied, would change the status of the underlying atom in the database. For instance, if the initial database is empty and all revision literals prescribed by revision rules are positive (i.e. of the form $\mathbf{in}(a)$), then justified weak revisions are guaranteed to be minimal and so, are justified revisions.

Computation and Complexity Results for Revision Programming. In this section we present the complexity of the basic reasoning task associated with revision programs: deciding the existence of a (weak) revision of a particular type.

Theorem 4. *Let $\mathcal{I}$ be a database and P a normal revision program. Then checking if there exists a P-justified revision (P-justified weak revision, respectively) for $\mathcal{I}$ is an NP-complete problem.* □

Theorem 5. *Let $\mathcal{I}$ be a database and P a revision program. Then checking if there exists a P-justified revision (P-justified weak revision, respectively) for $\mathcal{I}$ is a Σ_2^P-complete problem.* □

Theorem 6. *Let $\mathcal{I}$ be a database and P a revision program. Then checking if there exists a P-founded revision (P-founded weak revision, respectively) for $\mathcal{I}$ is a Σ_2^P-complete (NP-complete, respectively) problem.* □

These results show that the condition defining justified updates already makes the problem of the existence of a justified weak revision Σ_2^P-complete. Imposing, in addition, the minimality of change (considering justified revisions) does not increase the complexity. Foundedness is an "easier" condition. Deciding the existence of a founded weak revision is NP-complete. In this setting, imposing the minimality of change (switching to founded revisions) makes a difference. The complexity grows to Σ_2^P-complete.

Shifting theorem for revision programs. We will now study invariance under shifting [2]. Shifting consists of transforming an instance $\langle \mathcal{I}, P\rangle$ of the database repair problem to an isomorphic instance $\langle \mathcal{I}', P'\rangle$ by "shifting" $\mathcal{I}$ to $\mathcal{I}'$ and changing P to P' to reflect the "shift" of the database. A semantics for database revision has the *shifting property* if the revisions of the "shifted" instance $\langle \mathcal{I}', P'\rangle$ are precisely the results of modifying the revisions of the original instance $\langle \mathcal{I}, P\rangle$ according to the shift $\mathcal{I} \rightarrow \mathcal{I}'$. The shifting property is important. If it holds for a semantics, the study of that semantics can be reduced to the case when the input database is empty. Often, it allows us to relate revision programming and logic programming with negation.

Example 6. Let $\mathcal{I} = \{a, b\}$ and let $P = \{\mathbf{out}(a)|\mathbf{out}(b) \leftarrow\}$. There are two justified (and so, also founded) revisions for $\langle \mathcal{I}, P\rangle$: $\mathcal{E}_1 = \{\mathbf{out}(a)\}$ and $\mathcal{E}_2 = \{\mathbf{out}(b)\}$. Let $\mathcal{W} = \{a\}$. To "shift" the instance $\langle \mathcal{I}, P\rangle$ wrt $\mathcal{W}$, we first modify $\mathcal{I}$ by changing the status in $\mathcal{I}$ of elements in $\mathcal{W}$, in our case, of a. Since $a \in \mathcal{I}$, we remove it. Thus, $\mathcal{I}$ "shifted" wrt $\mathcal{W}$ becomes $\mathcal{J} = \{b\}$. Next, we modify P correspondingly, replacing revision literals involving a by their duals. That results in $P' = \{\mathbf{in}(a)|\mathbf{out}(b) \leftarrow\}$. The resulting instance $\langle \mathcal{J}, P'\rangle$ has two founded/justified revisions: $\{\mathbf{in}(a)\}$ and $\{\mathbf{out}(b)\}$. They can be obtained from the founded/justified revisions for $\langle \mathcal{I}, P\rangle$ by replacing $\mathbf{out}(a)$ with $\mathbf{in}(a)$ and $\mathbf{in}(a)$ with $\mathbf{out}(a)$ (the latter does not apply in this example). In other words, the original update problem and its shifted version are isomorphic. □

The situation presented in Example 6 is not coincidental. In this section we present results showing that the semantics of (weak) revisions, founded (weak) revisions and justified (weak) revisions satisfy the shifting property. We start by observing that *shifting* a database $\mathcal{I}$ to a database $\mathcal{I}'$ can be modeled by means of the symmetric difference operator. Namely, we have $\mathcal{I}' = \mathcal{I} \div \mathcal{W}$, where $\mathcal{W} = \mathcal{I} \div \mathcal{I}'$. This identity shows that one can shift any database $\mathcal{I}$ into any database $\mathcal{I}'$ by forming a symmetric difference of $\mathcal{I}$ with some set of atoms $\mathcal{W}$ (specifically, $\mathcal{W} = \mathcal{I} \div \mathcal{I}'$). We will now extend the operation of shifting a database wrt $\mathcal{W}$ to the case of revision literals and (resp. revision rules and revision programs). To this end, we introduce a *shifting* operator $T_{\mathcal{W}}$.

Definition 6. *Let $\mathcal{W}$ be a database and ℓ a revision literal. We define*

$$T_{\mathcal{W}}(\ell) = \begin{cases} \ell^D & \text{if the atom of } \ell \text{ is in } \mathcal{W} \\ \ell & \text{if the atom of } \ell \text{ is not in } \mathcal{W} \end{cases}$$

and we extend this definition to sets of revision literals. Furthermore, if op is an operator on sets of revision literals (such as conjunction or disjunction), for every set X of revision literals we define $T_{\mathcal{W}}(op(X)) = op(T_{\mathcal{W}}(X))$. Finally, for a revision rule $r = \alpha_1|\ldots|\alpha_k \leftarrow \beta_1, \ldots, \beta_m$, we set $T_{\mathcal{W}}(r) = T_{\mathcal{W}}(\alpha_1|\ldots|\alpha_k) \leftarrow T_{\mathcal{W}}(\beta_1, \ldots, \beta_m)$, and for a revision program P, $T_{\mathcal{W}}(P) = \{T_{\mathcal{W}}(r) \mid r \in P\}$.[2] □

Theorem 7. (SHIFTING THEOREM FOR REVISION PROGRAMS) *Let $\mathcal{I}$ and $\mathcal{W}$ be databases. For every revision program G and every consistent set $\mathcal{E}$ of revision literals:*

1. *$\mathcal{E}$ is a (weak) revision for $\mathcal{I}$ wrt G if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a (weak) revision for $\mathcal{I}$ wrt $T_{\mathcal{W}}(G)$*
2. *$\mathcal{E}$ is a G-justified (weak) revision for $\mathcal{I}$ if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a $T_{\mathcal{W}}(G)$-justified (weak) revision for $\mathcal{I}$*
3. *$\mathcal{E}$ is a G-founded (weak) revision for $\mathcal{I}$ if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a $T_{\mathcal{W}}(G)$-founded (weak) revision for $\mathcal{I}$*

5 Connections between Revision Programs and Active Integrity Constraints

We will now relate revision programs to active integrity constraints [9], an earlier formalism for expressing integrity constraints and preferred ways to enforce them.

Active integrity constraints — an overview. An *update action* is an expression $+a$ or $-a$, where $a \in At$. It states that a is to be inserted or deleted, respectively. A set $\mathcal{U}$ of update actions is *consistent* if it does not contain update actions $+a$ and $-a$, for any $a \in At$. For a database $\mathcal{D}$ and a consistent set $\mathcal{U}$ of update actions, we define the result of *updating $\mathcal{D}$ by means of $\mathcal{U}$* as the database

$$\mathcal{D} \circ \mathcal{U} = (\mathcal{D} \cup \{a \mid +a \in \mathcal{U}\}) \setminus \{a \mid -a \in \mathcal{U}\}.$$

An *integrity constraint* is a formula $r = L_1, \ldots, L_m \supset \bot$, where L_i, $1 \leq i \leq m$, are propositional literals (atoms a and their negations $not\ a$, for $a \in At$), and ',' stands for the conjunction. A database $\mathcal{D}$ *satisfies* an integrity constraint r, if $\mathcal{D}$ – viewed as a propositional interpretation – satisfies r. Given a set η of integrity constraints and a database $\mathcal{I}$, the problem of *database repair* is to update $\mathcal{I}$ with a set of update actions so that the resulting database satisfies all integrity constraints in η.

Definition 7. [WEAK REPAIRS AND REPAIRS]. *Let $\mathcal{I}$ be a database and η a set of integrity constraints. A* weak repair *for $\langle \mathcal{I}, \eta \rangle$ is a consistent set $\mathcal{U}$ of update actions such that $(\{+a \mid a \in \mathcal{I}\} \cup \{-a \mid a \in At \setminus \mathcal{I}\}) \cap \mathcal{U} = \emptyset$ ("essential" update actions only), and $\mathcal{I} \circ \mathcal{U} \models \eta$ (constraint enforcement).*

A consistent set $\mathcal{U}$ of update actions is a repair *for $\langle \mathcal{I}, \eta \rangle$ if it is a weak repair for $\langle \mathcal{I}, \eta \rangle$ and for every $\mathcal{U}' \subseteq \mathcal{U}$ such that $\mathcal{I} \circ \mathcal{U}' \models \eta$, $\mathcal{U}' = \mathcal{U}$ (minimality of change).* □

[2] We note that we overload the notation $T_{\mathcal{W}}$ and interpret it based on the type of the argument.

Let $r = a, b \supset \bot$, and let $\mathcal{I} = \{a, b\}$ be a database. Clearly, $\mathcal{I} \not\models r$. There are three possible weak repairs of $\mathcal{I}$ wrt r: $\{-a\}$, $\{-b\}$ and $\{-a, -b\}$. The first two are minimal and so, they are repairs. No weak repair and no repair is distinguished as preferred.

To model preferences on (weak) repairs, [9] modified the syntax of integrity constraints by allowing the user to list preferred update actions. We will now present that approach. For a propositional literal L, we write L^D for the dual literal to L. Further, for a literal $L = a$ ($L = not\ a$), we define $ua(L) = +a$ ($ua(L) = -a$). Similarly, for an update action $\alpha = +a$ ($\alpha = -a$), we define $lit(\alpha) = a$ ($lit(\alpha) = not\ a$). We call $+a$ and $-a$ the *duals* of each other, and write α^D to denote the dual of an update action α. We extend the notation introduced here to sets of literals and update actions. We define an *active integrity constraint* to be an expression of the form

$$r = L_1, \ \ldots, \ L_m \supset \alpha_1 | \ldots | \alpha_k \tag{2}$$

where $m, k \geq 0$, $m + k \geq 1$, L_i are literals, α_j are update actions, and

$$\{lit(\alpha_1)^D, \ldots, lit(\alpha_k)^D\} \subseteq \{L_1, \ldots, L_m\} \tag{3}$$

The set $\{L_1, \ldots, L_m\}$ is the *body* of r; we denote it by $body(r)$. Similarly, the set $\{\alpha_1, \ldots, \alpha_k\}$ is the *head* of r; we denote it by $head(r)$.

An active integrity constraint $L_1, \ \ldots, \ L_m \supset \alpha_1 | \ldots | \alpha_k$ represents the integrity constraint $L_1, \ \ldots, \ L_m \supset \bot$. Thus, we say that a database $\mathcal{D}$ *satisfies* an active integrity constraint r ($\mathcal{D} \models r$) if $\mathcal{D}$ satisfies the corresponding integrity constraint. In this way, the semantics of weak repairs and repairs lift to active integrity constraints. However, an active integrity constraint is more than an integrity constraint. It also specifies preferred update actions to use by listing them in the head.

The condition (3) ensures that an active integrity constraint supports only those update actions that pertain to its body and can fix it. It can be restated concisely as $[lit(head(r))]^D \subseteq body(r)$. We call literals in $[lit(head(r))]^D$ *updatable* by r, as they can be affected by an update action in $head(r)$. All other literals in $body(r)$ are *non-updatable* by r. We write $up(r)$ and $nup(r)$ for the sets of literals updatable and non-updatable by r, respectively.

Let $r = a, b \supset -b$ (cf. the previous example), and let $\mathcal{I} = \{a, b\}$ be a database. As before, $\mathcal{I} \not\models r$, and there are two repairs of $\mathcal{I}$ wrt r: $\{-a\}$ and $\{-b\}$. Now, since r lists $-b$ as an update action to execute, the latter one is preferred. The semantics of (weak) repairs are insensitive to such preferences. To reflect them, [9] defined *founded repairs*. A related concept of a founded weak repair was introduced in [8].

Definition 8. [FOUNDED (WEAK) REPAIRS]. *Let $\mathcal{I}$ be a database, η a set of active integrity constraints, and $\mathcal{U}$ a consistent set of update actions.*

1. *An update action α is* founded *wrt $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$ if there is $r \in \eta$ such that $\alpha \in head(r)$, $\mathcal{I} \circ \mathcal{U} \models nup(r)$, and $\mathcal{I} \circ \mathcal{U} \models \beta^D$, for every $\beta \in head(r) \setminus \{\alpha\}$.*
2. *The set $\mathcal{U}$ is* founded *wrt $\langle \mathcal{I}, \eta \rangle$ if every element of $\mathcal{U}$ is founded wrt $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$.*
3. *$\mathcal{U}$ is a* founded (weak) repair *for $\langle \mathcal{I}, \eta \rangle$ if $\mathcal{U}$ is a (weak) repair for $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$ is* founded *wrt $\langle \mathcal{I}, \eta \rangle$.* □

Foundedness captures a certain notion of "groundedness". Let us assume that $r \in \eta$ and $I \not\models r$. If α is founded wrt $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$ by means of r, then all literals in $body(r)$ other than $lit(\alpha^D)$ are satisfied by $\mathcal{I} \circ \mathcal{U}$. Thus, if $\mathcal{U}$ is to enforce r, it must contain α.

Founded repairs for $\langle \mathcal{I}, \eta \rangle$, despite being change-minimal and founded in $\langle \mathcal{I}, \eta \rangle$ ("grounded" in $\langle \mathcal{I}, \eta \rangle$) may still be self-justified. To address the problem [8] introduced justified (weak) repairs. To present the definition we need more terminology.

A set $\mathcal{U}$ of update actions is *closed* under an active integrity constraint r if $nup(r) \not\subseteq lit(\mathcal{U})$, or $head(r) \cap \mathcal{U} \neq \emptyset$. A set $\mathcal{U}$ of update actions is *closed* under a set η of active integrity constraints if it is closed under every $r \in \eta$. A *minimal* set closed under η can be viewed as "forced" by η, as all its elements are necessary (no nonempty subset can be dropped without violating the closedness condition).

Another concept we need is that of *no-effect actions*. Let $\mathcal{I}$ and $\mathcal{R}$ be databases. An update action $+a$ (respectively, $-a$) is a *no-effect* action for $(\mathcal{I}, \mathcal{R})$ if $a \in \mathcal{I} \cap \mathcal{R}$ (respectively, $a \notin \mathcal{I} \cup \mathcal{R}$). That is, a no-effect action does not change the status of its underlying atom. We denote by $ne(\mathcal{I}, \mathcal{R})$ the set of all no-effect actions wrt $(\mathcal{I}, \mathcal{R})$.

Definition 9. [JUSTIFIED (WEAK) REPAIR]. *Let $\mathcal{I}$ be a database, η a set of active integrity constraints, and $\mathcal{U}$ a consistent set of update actions.*

1. *$\mathcal{U}$ is a justified action set for $\langle \mathcal{I}, \eta \rangle$ if $\mathcal{U}$ is a minimal set of update actions containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ and closed under η.*
2. *If $\mathcal{U}$ is a justified action set for $\langle \mathcal{I}, \eta \rangle$, then $\mathcal{E} = \mathcal{U} \setminus ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is a* justified weak repair *for $\langle \mathcal{I}, \eta \rangle$. If in addition, for every $\mathcal{E}' \subseteq \mathcal{E}$ such that $\mathcal{I} \circ \mathcal{E}' \models \eta$, $\mathcal{E}' = \mathcal{E}$, then $\mathcal{E}$ is a* justified repair *for $\langle \mathcal{I}, \eta \rangle$.* □

Clearly, (founded, justified) weak repairs are (founded, justified) repairs. One can also show that justified (weak) repairs are founded (weak) repairs which, in turn, are (weak) repairs. There are no other inclusions between the six classes of weak repairs that we discussed here (cf. [9,8] for details).

Proper revision programs and a connection to active integrity constraints. There are striking similarities between the formalisms of revision programs and active integrity constraints. However, to relate the two, we first need to restrict the syntax of revision programs. We then show that the restriction does not change their expressivity.

A *proper revision rule* is a revision rule that satisfies the following condition: the literal in the head of the rule is not the dual of any literal in the body of the rule. A revision program is *proper* if all its revision rules are proper.

Theorem 8. *Let P be a revision program. There is a proper revision program P' such that for every database $\mathcal{I}$, (weak) revisions of $\mathcal{I}$ wrt P (P-founded (weak) revisions, P-justified (weak) revisions of $\mathcal{I}$, respectively) coincide with (weak) revisions of $\mathcal{I}$ wrt P' (P'-founded (weak) revisions, P'-justified (weak) revisions of $\mathcal{I}$, respectively).*

Given a proper revision rule r of the form

$$\alpha_1 | \ldots | \alpha_k \leftarrow \beta_1, \ldots \beta_m$$

we denote by $AIC(r)$ the active integrity constraint

$$lit(\beta_1), \ldots, lit(\beta_m), lit(\alpha_1)^D, \ldots, lit(\alpha_k)^D \supset ua(\alpha_1) | \ldots | ua(\alpha_k).$$

We note that if r is a revision constraint ($k = 0$), $AIC(r)$ is simply an integrity constraint. The operator $AIC(\cdot)$ is extended to proper revision programs in the standard way. It is easy to show that for each database $\mathcal{D}$, $\mathcal{D} \models P$ if and only if $\mathcal{D} \models AIC(P)$. We now have the following result.

Theorem 9. *Let P be a proper revision program. A set $\mathcal{E}$ of revision literals is a (weak) revision (resp. P-justified (weak) revision, P-founded (weak) revision) of $\mathcal{I}$ wrt P if and only if $ua(\mathcal{E})$ is a (weak) repair (resp. justified (weak) repair, founded (weak) repair) for $\langle \mathcal{I}, AIC(P)\rangle$.*

The mapping $AIC(\cdot)$ is a bijection between proper revision programs and sets of active integrity constraints. Thus, it establishes an *exact* bidirectional match between the two formalisms (for all semantics considered). We note that the restriction to proper programs is essential as $AIC(\cdot)$, when used with all revision programs, is no longer one-to-one and, in some cases, maps semantically different rules onto the same active integrity constraint. For instance, programs consisting of $\mathbf{in}(a) \leftarrow \mathbf{out}(a)$ and $\mathbf{in}(a) \leftarrow$, respectively, behave differently wrt $\mathcal{I} = \emptyset$ (the first program does not define any justified revisions, the second one does: $\mathcal{E} = \{\mathbf{in}(a)\}$). Yet, both rules are mapped by $AIC(\cdot)$ onto $not\ a \rightarrow +a$.

6 Discussion and Conclusions

We studied a formalisms of revision programming designed to support modeling of constraints on databases (belief sets), as well as preferred ways to enforce them if they are violated. Revision programming was proposed in [1,2] and further developed in [5]. However, the earlier work focused only on one semantics, the semantics of P-justified weak revisions. It is a limitation of the earlier work as, on the one hand, P-justified weak revisions do not satisfy the minimality of change property and, on the other, they may be too restrictive in situations when reasoning by cases may be justified.

Therefore, we proposed here several new declarative semantics for revision programs, by imposing the minimality of change property on P-justified revisions and/or by modifying the groundedness condition behind justified weak revisions. We studied properties of the resulting semantics. In particular, we established the complexity of several decision problems, and identified two classes of revision programs, for which justified weak revisions are change-minimal. Revision programming shows several similarities with the formalism of active integrity constraints [9]. We proposed an interpretation of revision rules as active integrity constraints and proved that under that interpretation, revision programming (restricted to proper revision programs) and the formalism of active integrity constrains are notational variants of each other.

Finally, we also proved that all semantics we studied satisfy the shifting property. We will now briefly point out that thanks to shifting, one can relate revision programs to Lifschitz-Woo programs [10], which generalize disjunctive logic programs by allowing the default negation in the heads of rules. Namely, *Lifschitz-Woo* programs consist of rules of the form

$$a_1 | \ldots | a_k | not\ b_1 | \ldots | not\ b_m \leftarrow c_1, \ldots, c_s, not\ d_1, \ldots not\ d_n \qquad (4)$$

where each a_i, b_i, c_i and d_i is an atom and not is a default negation. Given a revision rule

$$\mathbf{in}(a_1)|\ldots|\mathbf{in}(a_k)|\mathbf{out}(b_1)|\ldots|\mathbf{out}(b_m) \leftarrow \mathbf{in}(c_1),\ldots,\mathbf{in}(c_s),\mathbf{out}(d_1),\ldots\mathbf{out}(d_n) \tag{5}$$

there is a clear correspondence between the two. Let us denote by $LW(\cdot)$ a mapping that assigns a Lifschitz-Woo rule (4) to a revision rule (5). The following result was obtained in [5].

Theorem 10. *Let P be a revision program. Then, a set $\mathcal{U}$ of revision literals is a P-justified revision of $\emptyset$ if and only if $\mathcal{U} = \{\mathbf{in}(a) \mid a \in M\}$, where $M \subseteq At$ is a stable model of $LW(P)$ (according the the definition from [10]).*

Thanks to shifting, this result allows us to represent any revision program P and a database $\mathcal{I}$ as a Lifschitz-Woo program so that justified revisions correspond precisely to stable models, a property also observed in [5]. However, we proved here that shifting holds for other semantics of revision programs, too. Thus, Theorem 10 also suggests that *all* these semantics could be adapted directly to the setting of Lifschitz-Woo programs and, subsequently, to the formalism of programs with nested expressions [11] (subsuming Lifschitz-Woo programs). It follows that these generalized variants of logic programming can be endowed with a richer family of semantics that goes beyond the basic one given by stable models.

References

1. Marek, W., Truszczyński, M.: Revision specifications by means of programs. In: MacNish, C., Moniz Pereira, L., Pearce, D.J. (eds.) JELIA 1994. LNCS, vol. 838, pp. 122–136. Springer, Heidelberg (1994)
2. Marek, W., Truszczyński, M.: Revision programming. Theoretical Computer Science 190, 241–277 (1998)
3. Gelfond, M., Lifschitz, V.: The stable semantics for logic programs. In: Proceedings of ICLP 1988, pp. 1070–1080. MIT Press, Cambridge (1988)
4. Clark, K.: Negation as failure. In: Gallaire, H., Minker, J. (eds.) Logic and data bases, pp. 293–322. Plenum Press, New York (1978)
5. Pivkina, I.: Revision programming: a knowledge representation formalism. PhD thesis, Department of Computer Science, University of Kentucky (2001), http://lib.uky.edu/ETD/ukycosc2001d00022/pivkina.pdf
6. McCarthy, J.: Circumscription — a form of non-monotonic reasoning. Artificial Intelligence 13, 27–39 (1980)
7. Winslett, M.: Updating Logical Databases. Cambridge University Press, Cambridge (1990)
8. Caroprese, L., Truszczyński, M.: Declarative semantics for active integrity constraints (submitted, 2008), http://www.cs.uky.edu/ai/aic-full.pdf
9. Caroprese, L., Greco, S., Sirangelo, C., Zumpano, E.: Declarative semantics of production rules for integrity maintenance. In: Etalle, S., Truszczyński, M. (eds.) ICLP 2006. LNCS, vol. 4079, pp. 26–40. Springer, Heidelberg (2006)
10. Lifschitz, V., Woo, T.: Answer sets in general nonmonotonic reasoning. In: Proceedings of KR 1992, pp. 603–614. Morgan Kaufmann, San Francisco (1992)
11. Lifschitz, V., Tang, L.R., Turner, H.: Nested expressions in logic programs. Annals of Mathematics and Artificial Intelligence, 369–389 (1999)

Recovering Consistency by Forgetting Inconsistency

Sylvie Coste-Marquis and Pierre Marquis

Université Lille-Nord de France, Artois
CRIL UMR CNRS 8188, Lens, France
{coste,marquis}@cril.univ-artois.fr

Abstract. In this paper, we introduce and study a new paraconsistent inference relation $\models_c$ in the setting of 3-valued paraconsistent logics. Using inconsistency forgetting as a key mechanism for recovering consistency, it guarantees that the deductive closure $Cn_{\models_c}(\Sigma)$ of any belief base Σ is classically consistent and classically closed. This strong feature, not shared by previous inference relations in the same setting, allows to interpret an inconsistent belief base as a set of classical worlds (hence to reason classically from them).

1 Introduction

Reasoning in a non-trivial way from inconsistent pieces of information (the paraconsistency issue) is a fundamental problem in artificial intelligence. Its importance is reflected by the number of approaches developed so far to address it: paraconsistent logics, belief revision, belief merging, reasoning from preferred consistent subsets, knowledge integration, argumentative logics, purification, etc. (see [1,2,3] for a survey).

The variety of existing approaches can be explained by the fact that paraconsistency can be achieved in various ways, depending on the exact nature of the problem at hand (hence, the available information). Each of them has its own pros and cons, and is more or less suited to different inconsistency handling scenarios. For instance, when Σ represents the (conflicting) beliefs of several agents, a merged base giving the beliefs of the group of agents can be designed by logically weakening some local belief bases (associated to the agents) in order to restore global consistency [4,5,6,7,8].

Compared with the other approaches listed above, paraconsistent logics (taken *stricto sensu*) offer a *basic* way to address the trivialization issue in presence of inconsistency. Indeed, belief revision, belief merging, knowledge integration, reasoning from preferred consistent subsets and purification need some extra-logical information in order to be well-defined and avoid trivializing. Such extra-logical information can be rather poor: A splitting between the belief base and the revision formula in the belief revision setting, a set (or multi-set) organization of the beliefs in a belief merging scenario. They can also be rather sophisticated: Preference relations over beliefs, knowledge gathering actions for purification. In

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 113–125, 2008.

both cases, they are required. In particular, unlike paraconsistent logics, none of those approaches can address in a significant way the case when the available information take the form of a *single* piece (hence encoded as a unique formula in a logical language)[1].

Several (non mutually exclusive) techniques can be used to define an inference relation that avoid trivialization from an inconsistent propositional formula (see [3]). One of them consists in *preventing classically inconsistent belief bases from having no model*, through the consideration of more general notions of interpretations. Several multi-valued logics are related to this line of research (among others, see [10,11,12,13,14,15,16,17,18,19,20,21,22]).

In the following, the focus is laid on *three-valued paraconsistent logics*. The additional (epistemic) truth value (called middle element) intuitively means "proved both true and false" and allows to still reasoning meaningfully with variables that are not embedded directly in a contradiction. While a number of paraconsistent inference relations have been defined in this setting, none of them ensures that deductive closures are always classically consistent and classically closed. This is a strong drawback of such approaches since it prevents from interpreting inconsistent belief bases as sets of classical worlds (i.e., 2-interpretations), and consequently it questions the possibility to exploit further the information encoded by an inconsistent belief base using standard inference or decision-making techniques (since such techniques typically require classically consistent information).

In this paper, we fill the gap by introducing and studying a new paraconsistent inference relation $\models_c$ in the setting of three-valued paraconsistent logics. This inference relation elaborates on a valuable paraconsistent inference relation $\models^{\leq}$ introduced by Priest [15]. Basically, the preferred 3-models of a belief base Σ w.r.t. $\models_c$ are the 2-interpretations which are as close as possible to the preferred 3-models of a belief base Σ w.r.t. $\models^{\leq}$. Determining the latter models mainly amounts to forgetting the inconsistent "truth value" in the former interpretations. Interestingly, $\models_c$ guarantees that the deductive closure $Cn_{\models_c}(\Sigma)$ of any belief base Σ is classically consistent and classically closed (what we call the *classical closure property*).

The rest of this paper is organized as follows. In Section 2, we present some background on three-valued paraconsistent logics; especially, we define the logical framework into which our inference relation $\models_c$ takes place. In Section 3, we present the classical closure property and show that three-valued paraconsistent inference relations from the literature do not satisfy it. On this ground, we introduce our relation $\models_c$; we show that it satisfies a number of expected logical properties, including the strong paraconsistency condition (i.e., the deductive closure of a belief base never trivializes), the preservation property (i.e., the deductive closure of a belief base coincides with its classical closure when the

[1] Note that approaches based on consistent subsets take advantage of a specific "comma" connective [9] which is not equivalent to conjunction in the general case; every singleton consisting of an inconsistent formula like $\{a \wedge (\neg a \vee b) \wedge c \wedge \neg c\}$ has $\emptyset$ as its unique consistent subset.

belief base is classically consistent), as well as all the properties of system P [23] but reflexivity. We also investigate some computational aspects of $\models_c$, show that it is not harder than the underlying relation $\models^{\leq}$ from a complexity point of view and explain how to turn any finite belief base Σ into a consistent propositional formula $cl(\Sigma)$ such that $Cn_{\models_c}(\Sigma)$ is equal to the classical closure of $cl(\Sigma)$ (thus, $cl(\Sigma)$ can be viewed as a compilation of Σ as a propositional formula, classically interpreted). Finally, Section 4 concludes the paper. For space reasons, some proofs are omitted. However, they are given in [24], available from the authors.

2 Three-Valued Paraconsistent Logics

When a belief base is classically inconsistent, every formula is a classical consequence of it (*"ex falso quodlibet sequitur"*). In order to avoid such a trivialization, one can take advantage of any logic in which an (epistemic) truth value "both" ($\top$) denotes that a formula can be proved at the same time "true" (1) and "false" (0). This allows to highlight contradictory pieces of information, but still reasoning "reasonably" about the remaining ones. Thus the third truth value has to be understood as some encoding of the epistemic attitude "proved both true and false", and not as a standard truth value.

Now, there are a number of many-valued paraconsistent logics where such an (epistemic) truth value "both" is considered. In the following, we consider Kleene's strong three-valued logic with middle element designated, restricted to the so-called monotone fragment [18], i.e., the morphology of the language of the logic is reduced to the connectives $\neg$, $\vee$, $\wedge$, only. When restricted to this fragment, this logic is equivalent to a number of other logics pointed out so far in the literature, including LP [14], J_3 [10], $THREE$ [18] and other logics by Levesque [13] and Frisch [12].

Definition 1 (language). *$PROP_{PS}$ is the propositional language generated from a finite set PS of propositional symbols, the unary connective $\neg$ (negation) and the binary connectives $\vee$ (disjunction), and $\wedge$ (conjunction).*

Clearly, this language coincides with a standard language for classical propositional logic.

We will write propositional symbols a, b, ... and formulas from $PROP_{PS}$ will be denoted by lower case Greek letters α, β, ... Belief bases, that will be denoted by upper case Greek letters Σ,... are (conjunctively-interpreted) sets of formulas. In order to alleviate notations, we identify every singleton belief base $\{\alpha\}$ with the formula α in it. $Var(\Sigma)$ denotes the set of propositional symbols occurring in Σ.

A literal is a symbol $x \in PS$ or a negated one $\neg x$. x and $\neg x$ are said to be complementary literals. A proper subset of $PROP_{PS}$ is composed by the CNF formulas, i.e., the (finite) conjunctions of clauses, where a clause is a (finite) disjunction of literals. Another proper subset of $PROP_{PS}$ is composed by the DNF formulas, i.e., the (finite) disjunctions of terms, where a term is a (finite) conjunction of literals.

In the following, we consider a number of inference relations $\vdash$ over $PROP_{PS}$:

Definition 2 (inference relation)

- *An* inference relation $\vdash$ *is a subset of* $2^{PROP_{PS}} \times PROP_{PS}$.
- *For every* Σ *in* $2^{PROP_{PS}}$, $Cn_{\vdash}(\Sigma)$ *denotes the* deductive closure *of a set of formulas* Σ *w.r.t. the inference relation* $\vdash$, *i.e.,* $Cn_{\vdash}(\Sigma) = \{\alpha \in PROP_{PS} \mid \Sigma \vdash \alpha\}$.

We will also need the following notions of interpretations:

Definition 3 (interpretations)

- *A* 3-interpretation ω *over* $PROP_{PS}$ *is a total function from PS to* $\{0, 1, \top\}$.
- *A* 2-interpretation ω *over* $PROP_{PS}$ *is a total function from PS to* $\{0, 1\}$.

$3-\Omega$ (resp. $2-\Omega$) denotes the set of all 3-interpretations (resp. 2-interpretations). 2-interpretations are the classical worlds. Clearly, they are also 3-interpretations. However, the converse does not hold (we have $2 - \Omega \subset 3 - \Omega$).

In the logic under consideration, all the connectives are truth functional ones and the semantics $\omega(\alpha)$ of a formula α from $PROP_{PS}$ in a 3-interpretation ω is defined in the obvious compositional way given the following truth tables.

Table 1. Truth tables

α	β	$\neg\alpha$	$\alpha \wedge \beta$	$\alpha \vee \beta$
0	0	1	0	0
0	1	1	0	1
0	$\top$	1	0	$\top$
1	0	0	0	1
1	1	0	1	1
1	$\top$	0	$\top$	1
$\top$	0	$\top$	0	$\top$
$\top$	1	$\top$	$\top$	1
$\top$	$\top$	$\top$	$\top$	$\top$

It is easy to check that restricting the entries of the previous table to 0 and 1, one recovers the standard semantics for the connectives $\neg$, $\vee$, $\wedge$. Accordingly, a belief base can be considered classically unless it becomes inconsistent (typically via its expansion by a new, yet conflicting, piece of evidence).

In classical logic, notions of model and consequence are defined as:

Definition 4 ($\models_2$). *Let* ω *be a* 2*-interpretation over* $PROP_{PS}$. *Let* α *be a formula from* $PROP_{PS}$, *and let* Σ *be a set of formulas of* $PROP_{PS}$:

- ω *is a* 2-model *of* α *iff* $\omega(\alpha) = 1$.
- ω *is a* 2-model *of* Σ *iff* $\omega(\alpha) = 1$ *for every* $\alpha \in \Sigma$. $2 - mod(\Sigma)$ *denotes the set of* 2*-models of* Σ.
- α *is a* 2-consequence *of* Σ, *noted* $\Sigma \models_2 \alpha$, *iff every* 2*-model of* Σ *is a* 2*-model of* α.

A belief base Σ is *classsically consistent* iff it has a 2-model iff $Cn_{\models_2}(\Sigma) \neq PROP_{PS}$. It is well-known that $\models_2$ is not strongly paraconsistent:

Definition 5 (strong paraconsistency). *An inference relation* $\vdash$ *satisfies the* strong paraconsistency property *iff for every* Σ *in* $2^{PROP_{PS}}$, $Cn_{\vdash}(\Sigma) \neq PROP_{PS}$.

When dealing with more than two truth values, one has to make precise the set of designated values, i.e., the set of values that a formula can take to be considered as satisfied. Since we want to define a paraconsistent logic, we choose $\mathcal{D} = \{1, \top\}$: intuitively, a formula is satisfied if it is "at least true" (but it can also be false!). We are now ready to extend the previous notions of model and consequence to the three-valued case:

Definition 6 ($\models_3$). *Let* ω *be a* 3*-interpretation over* $PROP_{PS}$. *Let* α *be a formula from* $PROP_{PS}$, *and let* Σ *be a set of formulas of* $PROP_{PS}$:

- ω *is a* 3-model *of* α *iff* $\omega(\alpha) \in \mathcal{D}$.
- ω *is a* 3-model *of* Σ *iff* $\omega(\alpha) \in \mathcal{D}$ *for every* $\alpha \in \Sigma$. $3 - mod(\Sigma)$ *denotes the set of* 3*-models of* Σ.
- α *is a* 3-consequence *of* Σ, *noted* $\Sigma \models_3 \alpha$, *iff every* 3*-model of* Σ *is a* 3*-model of* α.

Two formulas α and β are said to be *strongly (*3*-)equivalent* iff for every 3-interpretation ω, we have $\omega(\alpha) = \omega(\beta)$.

Unlike $\models_2$, an interesting feature of the inference relation $\models_3$ is that it is strongly paraconsistent; indeed, every formula from $PROP_{PS}$ has a 3-model (the 3-interpretation $\omega_\top$ such that $\forall x \in PS$, $\omega_\top(x) = \top$). Thus, while we have $a \wedge \neg a \models_2 b$, we do *not* have $a \wedge \neg a \models_3 b$.

A problem is that $\models_3$ is a very weak inference relation. Especially, it is well-known that the disjunctive syllogism is not satisfied by $\models_3$: $a \wedge (\neg a \vee b) \not\models_3 b$. Thus, $\models_3$ does not satisfy the expected preservation property:

Definition 7 (preservation). *An inference relation* $\vdash$ *satisfies the* preservation property *iff for every* Σ *in* $2^{PROP_{PS}}$, *if* Σ *is classically consistent, then* $Cn_{\vdash}(\Sigma) = Cn_{\models_2}(\Sigma)$.

In order to circumvent this difficulty, other three-valued paraconsistent inference relations have been proposed. Some of them are based on the following principle: focus on some preferred models of Σ in order to keep as much information as possible. Thus, in LP_m [15], Priest suggests to prefer those 3-models of a belief base Σ which are "as classical as possible". Formally, let us consider the partial preordering $\leq$ over the set of 3-interpretations defined by $\omega \leq \omega'$ if and only if $\{x \in PS \mid \omega(x) = \top\} \subseteq \{x \in PS \mid \omega'(x) = \top\}$; the "most classical" 3-models of a belief base are the 3-models that are minimal w.r.t. $\leq$:

Definition 8 ($\models^{\leq}$). *Let* Σ *be a set of formulas of* $PROP_{PS}$. *Let* α *be a formula from* $PROP_{PS}$. $\Sigma \models^{\leq} \alpha$ *iff* $\forall \omega \in \min(3 - mod(\Sigma), \leq)$, $\omega(\alpha) \in \mathcal{D}$.

The resulting relation $\models^{\leq}$ is still strongly paraconsistent and it is strictly less cautious than $\models_3$, i.e., we have the inclusion $\models_3 \subset \models^{\leq}$. Unlike $\models_3$, it is non-monotonic; for instance, we have $a \wedge (\neg a \vee b) \models^{\leq} b$ but $a \wedge (\neg a \vee b) \wedge \neg a \not\models^{\leq} b$. Furthermore, $\models^{\leq}$ satisfies the preservation property: the preferred 3-models w.r.t. $\models^{\leq}$ of any classically consistent belief base Σ are exactly its 2-models.

Other inference relations have been defined so far for refining the inference relation $\models^{\leq}$ (especially in order to discriminate between the 3-consequences of a belief base Σ which are subject to a contradiction – like a if $\Sigma = a \wedge \neg a \wedge b$ – and those which are contradiction-free – like b if $\Sigma = a \wedge \neg a \wedge b$). Here are the main ones [20]:

Definition 9 (refined inference relations). *Let Σ be a set of formulas of $PROP_{PS}$. Let α be a formula from $PROP_{PS}$.*

- $\Sigma \models^{\leq}_{arg} \alpha$ *iff* $\Sigma \models^{\leq} \alpha$ *and* $\Sigma \not\models^{\leq} \neg\alpha$.
- $\Sigma \models^{\leq}_{1} \alpha$ *iff* $\forall \omega \in \min(3 - mod(\Sigma), \leq)$, $\omega(\alpha) = 1$.
- $\Sigma \models^{\leq}_{t} \alpha$ *iff* $\forall \omega \in \min(3 - mod(\Sigma), \leq)$, $\omega(\Sigma) \leq_t \omega(\alpha)$ *where the so-called "truth ordering" $\leq_t$ is such that* $0 \leq_t \top \leq_t 1$.

Those three relations correspond respectively to three refinement principles:

- Considering only argumentative consequences of the belief base.
- Selecting those consequences of the belief base that are "conflict-free" (i.e., true but not false).
- Selecting as consequences of the belief base formulas that are informally "more true" than the belief base.

All those relations are non-monotonic, strongly paraconsistent and they satisfy the preservation property. Furthermore they are strictly more cautious than $\models^{\leq}$ (see [20] for more details).

3 Recovering Consistency by Forgetting Inconsistency

3.1 The Inference Relation $\models_c$

Now, a major problem with the inference relations considered in the previous section (except $\models_2$ which is not paraconsistent) is that they do not satisfy the classical closure property:

Definition 10 (classical closure). *An inference relation $\vdash$ satisfies the* classical closure property *iff for every Σ in $2^{PROP_{PS}}$, $\vdash(\Sigma)$ is classically consistent and is closed w.r.t. classical deduction, i.e., $Cn_{\models_2}(Cn_{\vdash}(\Sigma)) = Cn_{\vdash}(\Sigma)$.*

This is obvious for $\models_3$, $\models^{\leq}$, and $\models^{\leq}_t$ since those relations are "reflexive" [20], i.e., for every α in $PROP_{PS}$, we have α is a consequence of α w.r.t. the relation. Thus, take $\Sigma = a \wedge \neg a$; Σ has to belong to its deductive closure w.r.t. any of those three relations, hence it cannot be classically consistent. As to $\models^{\leq}_{\mathrm{arg}}$, consider the

classically inconsistent CNF formula $\Sigma = (a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b) \wedge (\neg a \vee \neg b)$. Each of the four clauses in it is a consequence of Σ w.r.t. $\models^{\leq}_{\text{arg}}$: since their conjunction Σ is classically inconsistent, it cannot be the case that $Cn_{\models^{\leq}_{\text{arg}}}(\Sigma)$ is classically consistent and closed w.r.t. classical deduction. Finally, one can prove that $Cn_{\models^{\leq}_{t}}(\Sigma)$ is always classically consistent but this set is not necessarily closed w.r.t. classical deduction: take $\Sigma = a \wedge \neg a$; we have $\Sigma \not\models^{\leq}_{t} a \vee \neg a$. Since $a \vee \neg a$ is a classical tautology, the conclusion follows.

Using any of those inference relations thus prevents from interpreting inconsistent belief bases as sets of classical worlds (i.e., 2-interpretations), and consequently it questions the possibility to exploit further the information encoded by an inconsistent belief base using standard inference or decision-making techniques (since such techniques typically require classically consistent information). This motivates the introduction of our inference relation $\models_c$.

Intuitively, the preferred 3-models of a belief base Σ w.r.t. $\models_c$ are the 2-interpretations which are as close as possible to the preferred 3-models of a belief base Σ w.r.t. $\models^{\leq}$. Determining the latter models mainly amounts to forgetting the inconsistent "truth value" in the former interpretations. Formally, for any belief base Σ, we define $IncForg(\Sigma)$ as the set of 2-interpretations ω which are as close as possible to a 3-interpretation $\omega' \in \min(3 - mod(\Sigma), \leq)$, in the sense that $\forall x \in PS$, if $\omega'(x) \neq \top$, then $\omega'(x) = \omega(x)$. More formally, $IncForg(\Sigma) = \{\omega \in 2 - \Omega \mid \exists \omega' \in \min(3 - mod(\Sigma), \leq)\ \forall x \in PS$, if $\omega'(x) \neq \top$, then $\omega'(x) = \omega(x)\}$. Computing $IngForg(\Sigma)$ amounts to projecting each preferred 3-models of Σ on the variables classically interpreted in it (hence, forgetting inconsistency) and interpreting the resulting partial interpretations in a classical way. We are now ready to define $\models_c$:

Definition 11 ($\models_c$). *Let Σ be a set of formulas of $PROP_{PS}$. Let α be a formula from $PROP_{PS}$. $\Sigma \models_c \alpha$ iff $\forall \omega \in IncForg(\Sigma)$, $\omega(\alpha) = 1$.*

Example 1. Let $\Sigma = a \wedge (\neg a \vee b) \wedge c \wedge \neg c$. Assuming that $PS = \{a, b, c\}$, $\min(3 - mod(\Sigma), \leq)$ has only one preferred 3-model ω such that $\omega(a) = \omega(b) = 1$ and $\omega(c) = \top$. Accordingly, $IncForg(\Sigma)$ contains two elements ω' and ω'' such that $\omega'(a) = \omega'(b) = \omega''(a) = \omega''(b) = 1$ and $\omega'(c) = 0$ and $\omega''(c) = 1$. As a consequence, we have $\Sigma \models_c a \wedge b$, $\Sigma \not\models_c c$, and $\Sigma \not\models_c \neg c$. This contrasts with $\models^{\leq}$ which is such that $\Sigma \models^{\leq} c \wedge \neg c$.

Clearly enough, $\models_c$ is a non-monotonic inference relation. For instance, we have $a \models_c a$ but $a \wedge \neg a \not\models_c a$.

3.2 Logical Properties

We now investigate in more depth the logical properties satisfied by $\models_c$. Interestingly, $\models_c$ compares favourably with the underlying inference relation $\models^{\leq}$ w.r.t. logical properties: first of all, like $\models^{\leq}$, $\models_c$ also is strongly paraconsistent and satisfies the preservation property. Furthermore, it satisfies the classical closure property:

Proposition 1. $\models_c$ *is strongly paraconsistent and satisfies the preservation property and the classical closure property.*

Proof.
- *Strong paraconsistency:* Direct from the fact that $min(3-mod(\Sigma), \leq)$ is not empty whatever the belief base Σ, since this is the case for $3-mod(\Sigma)$ and $\leq$ is noetherian since PS is finite.
- *Preservation:* If Σ is classically consistent, then $min(3-mod(\Sigma), \leq) = 2-mod(\Sigma)$. Consequently, $IncForg(\Sigma) = 2-mod(\Sigma)$, conclusion follows.
- *Classical closure:* Since $min(3-mod(\Sigma), \leq)$ is not empty (see above), this is also the case of $IncForg(\Sigma)$. Hence $\models_c(\Sigma)$ is classically consistent. Since $IncForg(\Sigma) \subseteq 2-\Omega$, we obviously have that $\models_c(\Sigma)$ is closed w.r.t. classical deduction: $\models_2(\models_c(\Sigma)) = \models_c(\Sigma)$. □

Now, compared with $\models^{\leq}_{\text{arg}}$, $\models^{\leq}_{1}$ and $\models^{\leq}_{t}$, $\models^{\leq}$ exhibits quite a good logical behaviour in the sense that it is a preferential inference relation [20]:

Definition 12 (system P). *An inference relation* $\vdash$ *is* preferential *iff it satisfies the following properties (system* P*):*

(Ref)	$\alpha \vdash \alpha$	Reflexivity
(LLE)	If α and β are strongly 3-equivalent and $\alpha \vdash \gamma$, then $\beta \vdash \gamma$	Left Logical Equivalence
(RW)	If $\alpha \vdash \beta$ and $\beta \models_3 \gamma$, then $\alpha \vdash \gamma$	Right Weakening
(Or)	If $\alpha \vdash \gamma$ and $\beta \vdash \gamma$, then $\alpha \vee \beta \vdash \gamma$	Or
(Cut)	If $\alpha \wedge \beta \vdash \gamma$ and $\alpha \vdash \beta$, then $\alpha \vdash \gamma$	Cut
(CM)	If $\alpha \vdash \beta$ and $\alpha \vdash \gamma$, then $\alpha \wedge \beta \vdash \gamma$	Cautious Monotony

Following seminal works in non-monotonic logic [25,26,23,27], this set of normative properties that a non-monotonic inference relation should satisfy has been given in [23]. These properties have been primarily stated in the framework of classical logic [23], but they can be extended to multi-valued settings in a straightforward way as above (such an extension has also been considered in [18]).

Thus, an important question is to determine whether going from $\models^{\leq}$ to $\models_c$ leads to lose such valuable logical properties. Fortunately, most important properties still hold but reflexivity:

Proposition 2. $\models_c$ *satisfies all the properties of system* P*, except reflexivity.*

Proof.
- *Reflexivity:* Take $\alpha = a \wedge \neg a$. We have $\alpha \not\models_c \alpha$.
- *Left Logical Equivalence:* Obvious from the fact that (strongly) equivalent formulas have the same 3-models.
- *Right Weakening:* If $\beta \models_3 \gamma$, then $\beta \models_2 \gamma$ due to the inclusion $2-mod(\beta) \subseteq 3-mod(\beta)$. The fact that $\models_c$ satisfies the classical closure property concludes the proof.
- *Or:* We have that $3-mod(\alpha \vee \beta) = 3-mod(\alpha) \cup 3-mod(\beta)$. As a consequence, $min(3-mod(\alpha \vee \beta), \leq) \subseteq min(3-mod(\alpha), \leq) \cup min(3-mod(\beta), \leq)$. Therefore, $IncForg(\alpha \vee \beta) \subseteq IncForg(\alpha) \cup IncForg(\beta)$. Since every $\omega \in IncForg(\alpha) \cup IncForg(\beta)$ is such that $\omega(\gamma) = 1$ when $\alpha \vdash \gamma$ and $\beta \vdash \gamma$, this must be the case for every $\omega \in IncForg(\alpha \vee \beta)$.
- *Cut:* We first prove the following lemma:

Lemma 1. *Let ω and ω' be two 3-interpretations such that $\forall x \in PS$, if $\omega'(x) \neq \top$, then $\omega'(x) = \omega(x)$. Then for any formula α of $PROP_{PS}$, we have that if $\omega'(\alpha) = 1$ (resp. $\omega'(\alpha) = 0$), then $\omega(\alpha) = 1$ (resp. $\omega(\alpha) = 0$).*

The proof of this lemma is easy by structural induction on α. Now, by *reductio ad absurdum*, assume that there exists $\omega \in IncForg(\alpha)$ such that $\omega(\gamma) = 0$. Then by definition of $IncForg(\alpha)$, there exists $\omega' \in min(3 - mod(\alpha), \leq)$ such that $\forall x \in PS$, if $\omega'(x) \neq \top$, then $\omega'(x) = \omega(x)$. Since $\omega' \in 3 - mod(\alpha)$, we have that $\omega'(\alpha) \neq 0$. Since $\alpha \models_c \beta$, we have that $\omega(\beta) = 1$. As a consequence of the lemma, we get that $\omega'(\beta) \neq 0$. Hence, we have $\omega'(\alpha \wedge \beta) \neq 0$: $\omega' \in 3 - mod(\alpha \wedge \beta)$. Since $3 - mod(\alpha \wedge \beta) \subseteq 3 - mod(\alpha)$ and $\omega' \in min(3 - mod(\alpha), \leq)$, we must have $\omega' \in min(3 - mod(\alpha \wedge \beta), \leq)$. Hence $\omega \in IncForg(\alpha \wedge \beta)$. Since $\alpha \wedge \beta \models_c \gamma$, we must have $\omega(\gamma) = 1$, contradiction.

– *Cautious Monotony:* We first exploit the previous lemma to show that for any formulas α and β of $PROP_{PS}$, if $\alpha \models_c \beta$, then $\alpha \models^{\leq} \beta$. By *reduction ad absurdum*, assume that there exists $\omega' \in min(3 - mod(\alpha), \leq)$ such that $\omega'(\beta) = 0$. From the lemma, for every 2-interpretation ω that $\forall x \in PS$, if $\omega'(x) \neq \top$, then $\omega'(x) = \omega(x)$, we must have $\omega(\beta) = 0$. Since $\omega' \in min(3 - mod(\alpha), \leq)$, for at least one 2-interpretation $\omega \in IncForg(\alpha)$, we must have $\omega(\alpha) = 0$. This contradicts the fact that $\alpha \models_c \beta$.

  Now, in order to prove the Cautious Monotony property, it is enough to show that whenever $\alpha \models_c \beta$, we have that $min(3 - mod(\alpha \wedge \beta), \leq) = min(3 - mod(\alpha), \leq)$. Let $\omega \in min(3 - mod(\alpha), \leq)$. Since $\alpha \models_c \beta$, we have that $\alpha \models^{\leq} \beta$. Hence, ω is a 3-model of β. Since it is a 3-model of α, it is a 3-model of $\alpha \wedge \beta$. Since $3 - mod(\alpha \wedge \beta) \subseteq 3 - mod(\alpha)$, we have that $\omega \in min(3 - mod(\alpha \wedge \beta), \leq)$. Hence the inclusion $min(3 - mod(\alpha), \leq) \subseteq min(3 - mod(\alpha \wedge \beta), \leq)$ holds. Conversely, assume that there exists $\omega' \in min(3 - mod(\alpha \wedge \beta), \leq) \setminus min(3 - mod(\alpha), \leq)$. Since $3 - mod(\alpha \wedge \beta) \subseteq 3 - mod(\alpha)$, there exists $\omega \in min(3 - mod(\alpha), \leq)$ such that $\omega < \omega'$ (i.e., $\omega \leq \omega'$ and $\omega' \not\leq \omega$). From the previous inclusion, we must have that $\omega \in min(3 - mod(\alpha \wedge \beta), \leq)$. The fact that $\omega < \omega'$ contradicts that $\omega' \in min(3 - mod(\alpha \wedge \beta), \leq)$. □

Observe that there would be no way to keep reflexivity while ensuring the classical closure property. Indeed, we have the following easy proposition:

Proposition 3. *No inference relation $\vdash$ satisfies both reflexivity and the classical closure property.*

Proof. Consider $\Sigma = a \wedge \neg a$. If $\Sigma \nvdash \Sigma$ then it does not satisfy reflexivity. Contrastingly, If $\Sigma \vdash \Sigma$ then it does not satisfy the classical closure property since Σ is classically inconsistent. □

It is also interesting to note that $\models_c$ satisfies other properties which are not shared by $\models^{\leq}$ [20], especially "transitivity" (this is a direct consequence of the fact that it satisfies both the classical closure property and the preservation property):

Proposition 4. *$\models_c$ satisfies transitivity, i.e. for any formulas α, β, γ from $PROP_{PS}$, if $\alpha \models_c \beta$ and $\beta \models_c \gamma$, then $\alpha \models_c \gamma$.*

Finally, it is important to determine whether the relaxation of $\models^{\leq}$ we realised to ensure the classical closure property does not lead to a too weak inference relation $\models_c$. The following inclusions show that this is not the case:

Proposition 5. $\models_1^{\leq} \subset \models_c \subset \models^{\leq}$.

Thus, all the "conflict-free" consequences α of a belief base Σ w.r.t. $\models^{\leq}$ are preserved by $\models_c$. Furthermore, $\models_c$ does not add consequences that would not be derivable using $\models_1^{\leq}$.

3.3 Computational Aspects

In this section, we investigate some computational aspects of $\models_c$. We assume the reader familiar with some basic notions of complexity, especially the complexity classes coNP and Π_2^p of the polynomial hierarchy PH (see [28] for a survey).

We first consider the complexity of the inference problem for $\models_c$:

Definition 13 ($\models_c$-INFERENCE). $\models_c$-INFERENCE *is the following decision problem:*

- **Input:** *A finite set Σ of formulas from $PROP_{PS}$ and a formula α in $PROP_{PS}$.*
- **Question:** *Does $\Sigma \models_c \alpha$ hold?*

We have obtained the following result:

Proposition 6. $\models_c$-INFERENCE *is Π_2^p-complete.*

Proof. Membership is easy; one considers the complementary problem: in order to show that $\Sigma \models_c \alpha$ holds, we guess a 2-interpretation ω and a 3-interpretation ω' over $Var(\Sigma) \cup Var(\alpha)$; then we check that ω' belongs to $min(3 - mod(\Sigma), \leq)$ (one call to an NP oracle since this problem is in coNP); finally, we check in polynomial time that for every $x \in Var(\Sigma) \cup Var(\alpha)$, we have that $\omega(x) = \omega'(x)$ whenever $\omega'(x) \neq \top$, and that $\omega(\alpha) = 1$.

Hardness holds even in the restricted case when Σ is a CNF formula and α is a propositional symbol; we consider the problem of determining, given a CNF formula Σ and a symbol a, whether every element ω of $min(3 - mod(\Sigma), \leq)$ is such that $\omega(a) \neq \top$. This problem has been shown Π_2^p-hard in [22]. The fact that every element ω of $min(3 - mod(\Sigma), \leq)$ is such that $\omega(a) \neq \top$ if and only if $\Sigma \wedge (a \vee b) \wedge (\neg a \vee b) \models_c b$ where $b \in PS \setminus Var(\Sigma)$, completes the proof. □

This proposition shows that $\models_c$ is not harder than the underlying relation $\models^{\leq}$ from a computational complexity point of view; indeed, the inference problem for $\models^{\leq}$ also is Π_2^p-complete [22].

We now show how to turn any finite belief base Σ (viewed as the conjunction of its elements) into a "classical" consistent propositional formula $cl(\Sigma)$ such that $\models_c (\Sigma)$ is equal to the classical closure of $cl(\Sigma)$. The basic idea is to turn first Σ into a DNF formula which is strongly equivalent. As in classical propositional logic, such a DNF formula can be computed by applying iteratively to Σ the following equivalences, considered as rewrite rules (left-to-right oriented):

- $\neg(\neg\alpha)$ is strongly 3-equivalent to α.
- $\neg(\alpha \vee \beta)$ is strongly 3-equivalent to $(\neg\alpha) \wedge (\neg\beta)$.
- $\neg(\alpha \wedge \beta)$ is strongly 3-equivalent to $(\neg\alpha) \vee (\neg\beta)$.
- $\alpha \wedge (\beta \vee \gamma)$ is strongly 3-equivalent to $(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$ (and similarly for $(\beta \vee \gamma) \wedge \alpha$).

Of course, the obtained DNF formula can be of exponential size in the size of Σ. It now remains to forget inconsistencies in this DNF formula after isolating terms representing the preferred models (the minimization step); formally, for every term α, let $inc(\alpha)$ be the set of "inconsistencies" occurring in α: $inc(\alpha) = \{x \in PS \mid x \text{ and } \neg x \text{ occur in } \alpha\}$. $cl(\Sigma)$ is the DNF formula obtained by successively:

1. Removing in the current DNF every term α such that $inc(\alpha)$ is not minimal w.r.t. set-inclusion in the set $\{inc(\alpha) \mid \alpha \text{ a term in the current DNF}\}$.
2. Removing in every term of the resulting DNF formula every literal l when the complementary literal also occurs in the term, then removing every empty term (and finally adding $a \vee \neg a$ if the resulting DNF formula contains no term).

We have that:

Proposition 7. $Cn_{\models_c}(\Sigma) = Cn_{\models_2}(cl(\Sigma))$.

As a matter of illustration, consider again Example 1: let $\Sigma = a \wedge (\neg a \vee b) \wedge c \wedge \neg c$ and $PS = \{a, b, c\}$. Σ is strongly 3-equivalent to the following DNF formula $(a \wedge \neg a \wedge c \wedge \neg c) \vee (a \wedge b \wedge c \wedge \neg c)$. Now, forgetting inconsistency in Σ leads to the DNF formula $cl(\Sigma) = a \wedge b$ (the first term $(a \wedge \neg a \wedge c \wedge \neg c)$ of the previous DNF is removed during the minimization step). We can easily check that $Cn_{\models_c}(\Sigma) = Cn_{\models_2}(cl(\Sigma))$.

Since the computation of $cl(\Sigma)$ can be achieved in time polynomial in the size of Σ when Σ is a DNF and since $cl(\Sigma)$ is a DNF formula, we easily get that:

Proposition 8

- *Under the restriction where Σ is a DNF formula, $\models_c$-*INFERENCE *is* coNP*-complete.*
- *Under the restriction where Σ is a DNF formula and α is a CNF formula, $\models_c$-*INFERENCE *is in* P*.*

Thus the formula $cl(\Sigma)$ is a classically consistent formula which can be viewed as a compilation of Σ (in the sense that any finite belief base Σ interpreted w.r.t. $\models_c$ is equivalent to the corresponding formula $cl(\Sigma)$ classically interpreted and that the inference problem from $cl(\Sigma)$ is computationally easier than the inference problem from Σ, unless the polynomial hierarchy collapses at the first level).

4 Conclusion

In this paper, we have introduced and studied a new paraconsistent inference relation $\models_c$ in the setting of 3-valued paraconsistent logics. Using inconsistency

forgetting as a key mechanism for recovering consistency, it guarantees that the deductive closure $Cn_{\models_c}(\Sigma)$ of any belief base Σ is classically consistent and classically closed. This strong feature, not shared by previous inference relations in the same setting, allows to interpret an inconsistent belief base as a set of classical worlds (hence to reason classically from them).We have investigated the logical properties and the computational complexity of $\models_c$. Among other things, we have shown that $\models_c$ satisfies many interesting properties which are shared by the underlying inference relation $\models^{\leq}$, without any complexity shift compared to it.

We have considered in this paper a basic language for three-valued paraconsistent logic (the monotone fragment). A first perspective for further research is to extend the approach to more complex morphologies. It is also clear that the inconsistency forgetting mechanism at work here could be applied to other many-valued paraconsistent logics, especially four-valued ones. This is another extension of this work that we plan to do.

Acknowledgements

The authors want to thank the anonymous referees for their helpful comments. They have been partly supported by the IUT de Lens.

References

1. Besnard, P., Hunter, A.: Introduction to actual and potential contradictions. In: Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol. 2, pp. 1–11. Kluwer Academic Publishers, Dordrecht (1998)
2. Hunter, A.: Paraconsistent logics. In: Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol. 2, pp. 11–36. Kluwer Academic Publishers, Dordrecht (1998)
3. Priest, G.: Paraconsistent Logic. In: Handbook of Philosophical Logic, vol. 6, pp. 287–393. Kluwer Academic Publishers, Dordrecht (2002)
4. Grant, J., Subrahmanian, V.: Reasoning in inconsistent knowledge bases. IEEE Trans. on Knowledge and Data Engineering 7(1), 177–189 (1995)
5. Lin, J.: Integration of weighted knowledge bases. Artificial Intelligence 83(2), 363–378 (1996)
6. Revesz, P.Z.: On the semantics of arbitration. International Journal of Algebra and Computation 7(2), 133–160 (1997)
7. Konieczny, S., Pino Pérez, R.: On the logic of merging. In: Proc. of KR 1998, Trento, Italy, pp. 488–498 (1998)
8. Konieczny, S.: On the difference between merging knowledge bases and combining them. In: Proc. of KR 2000, Breckenridge, CO, pp. 135–144 (2000)
9. Konieczny, S., Lang, J., Marquis, P.: Reasoning under inconsistency: The forgotten connective. In: Proc. of IJCAI 2005, Edinburgh, UK, pp. 484–489 (2005)
10. D'Ottaviano, I., da Costa, N.: Sur un problème de Jaśkowski. Comptes Rendus de l'Académie des Sciences de Paris 270, 1349–1353 (1970)
11. Belnap, N.: A useful four–valued logics. In: Modern Uses of Multiple-Valued Logic, pp. 8–37. Reidel (1977)

12. Frisch, A.: Inference without chaining. In: Proc. of IJCAI 1987, Milan, Italy, pp. 515–519 (1987)
13. Levesque, H.: A knowledge-level account of abduction (preliminary version). In: Proc. of IJCAI 1989, Detroit, MI, pp. 1061–1067 (1989)
14. Priest, G.: Reasoning about truth. Artificial Intelligence 39, 231–244 (1989)
15. Priest, G.: Minimally inconsistent LP. Studia Logica 50, 321–331 (1991)
16. Besnard, P., Schaub, T.: Circumscribing inconsistency. In: Proc. of IJCAI 1997, Nagoya, Japan, pp. 150–155 (1997)
17. Besnard, P., Schaub, T.: Signed systems for paraconsistent reasoning. J. of Automated Reasoning 20, 191–213 (1998)
18. Arieli, O., Avron, A.: The value of four values. Artificial Intelligence 102, 97–141 (1998)
19. Arieli, O., Avron, A.: A model-theoretic approach for recovering consistent data from inconsistent knowledge bases. J. of Automated Reasoning 22(2), 263–309 (1999)
20. Konieczny, S., Marquis, P.: Three-valued logics for inconsistency handling. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 332–344. Springer, Heidelberg (2002)
21. Marquis, P., Porquet, N.: Resource-bounded paraconsistent inference. Ann. of Mathematics and Artificial Intelligence 39, 349–384 (2003)
22. Coste-Marquis, S., Marquis, P.: On the complexity of paraconsistent inference relations. In: Bertossi, L., Hunter, A., Schaub, T. (eds.) Inconsistency Tolerance. LNCS, vol. 3300, pp. 151–190. Springer, Heidelberg (2005)
23. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
24. Coste-Marquis, S., Marquis, P.: Recovering consistency by forgetting inconsistency. Technical report, CRIL UMR 8188 (2008)
25. Gabbay, D.M.: Theoretical foundations for nonmonotonic reasoning in experts systems. In: Apt, K. (ed.) Logic and Models of Concurrent Systems. Springer, Heidelberg (1985)
26. Makinson, D.: General Pattern in nonmonotonic reasoning. In: Handbook of Logic in Artificial Intelligence and Logic Programming, vol. III, pp. 35–110. Clarendon Press, Oxford (1994)
27. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55, 1–60 (1992)
28. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading (1994)

On the Credal Structure of Consistent Probabilities

Fabio Cuzzolin

Oxford Brookes University
fabiocuzzolin@gmail.com

Abstract. In this paper we introduce a novel, simpler form of the polytope of inner Bayesian approximations of a belief function, or "consistent probabilities". We prove that the set of vertices of this polytope is generated by all possible permutations of elements of the domain, mirroring a similar behavior of outer consonant approximations.

1 Introduction

Uncertainty description [1] is a composite field in which different but related approaches compete to gain a wider audience in engineering [2] and business [3] applications. Belief [4], probability, and possibility [5] measures can all be used to represent uncertainty, even though some of them may be more fit to specific domains of applications. Their relation is then a popular object of study. More specifically, it is interesting in several situations to pose the problem of transforming one uncertainty measure into a measure of a different class. In the case of belief functions (b.f.s), this issue goes under the name of "Bayesian" or "consonant" approximation problem, according to whether we seek to approximate a belief measure with a probability or a possibility.

In particular we may request the approximating probabilities to be more informative of the original belief function $b : 2^\Theta \rightarrow [0, 1]$ (where Θ is a finite set, and b maps subsets $A \subseteq \Theta$ of Θ to $[0, 1]$). The "least commitment principle" [6] postulates indeed that, given a set of measures compatible with a number of constraints, the most appropriate one is the "least informative".

However, there are many ways of measuring the information content of a belief function [7,8,9]. If we adopt the classical ordering $b' \geq b \equiv b'(A) \geq b(A)$ $\forall A \subseteq \Theta$, we obtain the set of *inner Bayesian approximations* $\mathcal{P}[b]$ of b, i.e. the probabilities whose values dominates that of b on all events:

$$\mathcal{P}[b] = \{p \in \mathcal{P} : b(A) \leq p(A) \; \forall A \subseteq \Theta\} \tag{1}$$

($\mathcal{P}$ denoting the set of all probability measures on Θ).

According to Equation (1) this "credal set" [10] of probability distributions is in fact the set of probabilities which admit the original b.f. b as a lower bound. A powerful semantics to belief functions comes from the remark that such probabilities can be obtained by redistributing the basic belief or "mass"

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 126–139, 2008.

of each event $A \subseteq \Theta$ to the elements it contains. Belief functions can indeed be seen as constraints on the probability simplex, where they define the credal set (1).

Geometrically, inner Bayesian approximations or *consistent probabilities* are known to form a *polytope* (the convex closure of a finite number of points) in the space of probability measures, whose center of mass coincides with the pignistic transformation [11,12]. The credal semantics of belief functions is central in the "Transferable Belief Model" [13,14,15,16]. There belief is represented at credal level, while decision are made by recurring to the pignistic transformation. In robust Bayesian statistics, more in general, a large literature exists on the study of convex sets of probability distributions [17,18,19,20].

The goal of this paper is to prove a new result on the geometry of consistent probabilities, which greatly simplifies its classical expression.

We prove that the set of actual vertices of the polytope $\mathcal{P}[b]$ is indeed quite small, and determined by all possible permutations of elements of the domain:

$$\mathcal{P}[b] = Cl(p^\rho[b] \; \forall \rho)$$

where $p^\rho[b]$ is a probability determined by a permutation $\rho = \{x_{\rho(1)}, ..., x_{\rho(n)}\}$ of the singletons of Θ. This generates a beautiful symmetry with the (dual) case of *outer consonant approximations* [21,22], i.e. the consonant b.f.s dominated by b: $\mathcal{O}[b] = \left\{co \in \mathcal{CO} : co(A) \leq b(A) \; \forall A \subseteq \Theta\right\}$ (here $\mathcal{CO}$ denotes the collection of all consonant b.f.s, i.e. belief functions whose focal elements are nested [4]).

As for each maximal chain of focal elements a vertex of the polytope of outer consonant approximations is also determined by a permutation of singletons, there exists a 1-1 correspondence between actual vertices of $\mathcal{P}[b]$ and $\mathcal{O}[b]$.

Paper outline. We recall in Section 2 the interpretation of b.f.s as lower bounds to a convex set or polytope of "consistent" probabilities. In Section 3 we prove that the actual vertices of this polytope are each associated with a permutation of the elements of the domain, and discuss their uniqueness. These vertices are in 1-1 correspondence with the vertices of the region of outer consonant approximations induced by singleton permutations (Section 4).

2 Probabilities Consistent with a Belief Function

Belief [4], probability, and possibility [5] theory are different but related descriptions of uncertainty, as (at least in the finite setting) both probabilities and possibilities are special cases of b.f.s. If we suppose that the ideal knowledge state is represented by a "true", but unknown probability measure (which we cannot estimate precisely because of imprecise measurements, missing data, etcetera) belief measures have in turn a natural interpretation as lower/upper bounds to this unknown true probability. Each belief function is then naturally associated with the set of probabilities which actually meet these constraints.

Belief measures. Belief functions are mathematical representations of the bodies of evidence we possess on a given decision or estimation problem Q. We assume that the possible answers to Q form a finite set $\Theta = \{x_1, ..., x_n\}$ called

"frame of discernment". A *basic probability assignment* (b.p.a.) [4] over Θ is a function $m : 2^\Theta \to [0,1]$ on its power set $2^\Theta = \{A \subseteq \Theta\}$ such that: 1. $m(\emptyset) = 0$; 2. $\sum_{A \subseteq \Theta} m(A) = 1$; 3. $m(A) \geq 0 \; \forall A \subseteq \Theta$. Subsets of Θ associated with non-zero values of m are called "focal elements" (f.e.s).

The *belief function* $b : 2^\Theta \to [0,1]$ associated with a b.p.a. m_b on Θ is defined as

$$b(A) = \sum_{B \subseteq A} m_b(B). \tag{2}$$

In the following we will denote by b_A the "dogmatic" b.f. which assigns unitary mass to a single event A: $m_b(A) = 1$, $m_b(B) = 0 \; \forall B \neq A$. We can then write each belief function b with b.p.a. $m_b(A)$ as [23]

$$b = \sum_{A \subseteq \Theta} m_b(A) b_A. \tag{3}$$

A finite probability is a special b.f. assigning non-zero masses to singletons only (*Bayesian* b.f.): $m_b(A) = 0 \; |A| > 1$. The *plausibility function* (pl.f.) $pl_b : 2^\Theta \to [0,1]$, $pl_b(A) \doteq 1 - b(A^c)$ measures the amount of evidence *not against* A.

Consistent probabilities. Even though originally defined as set functions of the form (2) on the power set of a finite universe [4], belief functions have a natural interpretation as constraints on the "true", unknown probability distribution describing a state of belief.

Given a certain amount of evidence we are allowed to describe our belief on the outcome of Q in several possible ways: the classical option is to assume a probability distribution on Θ. However, as we may need to incorporate imprecise measurements and people's opinions in our knowledge state, or cope with missing or scarce information, a more sensible approach is to assume that we have no access to the "correct" probability distribution but that the available evidence provides us with some sort of constraint on this true distribution. Belief functions are mathematical descriptions of such a constraint.

According to this interpretation a f.e. A of mass $m_b(A)$ can be seen as the indication of the existence of a mass $m_b(A)$ "floating" inside A. The mass assigned to each event $A \subseteq \Theta$ can float freely among its elements $x \in A$. A probability distribution compatible with b emerges by redistributing the mass of each focal element to its singletons. This set of Bayesian b.f.s is said "consistent" with b.

Example. To illustrate the notion of probability consistent with a belief function let us consider a little toy example, namely a b.f. b on a frame of cardinality three $\Theta = \{x, y, z\}$ with focal elements (Figure 1-a)

$$m_b(\{x, y\}) = \frac{2}{3}, \qquad m_b(\{y, z\}) = \frac{1}{3}. \tag{4}$$

We can build a probability consistent with b by, for instance, equally sharing the mass of $\{x, y\}$ among its elements x and y, while attributing the entire mass of

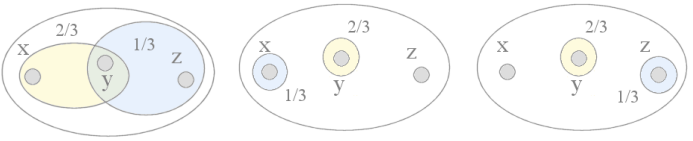


Fig. 1. a) A simple belief function in the ternary frame. b) and c) two admissible probabilities consistent with a).

$\{y, z\}$ to y (Figure 1-b). Or, we can assign all the mass of $\{x, y\}$ to y, and give the mass of $\{y, z\}$ to z only, obtaining the Bayesian belief function of Figure 1-c.

Belief functions as lower bounds. An alternative vision of the set of probabilities consistent with a b.f. b relates to the fact that the set of all and only the (admissible) probabilities obtained as above by re-assigning the mass of each f.e. to its elements is (1) $\mathcal{P}[b] \doteq \{p \in \mathcal{P} : p(A) \geq b(A)\ \forall A \subseteq \Theta\}$ i.e. the distributions whose values dominate that of b on all events A.

A b.f. can then be interpreted as a "lower bound" to the set of probabilities it determines. For instance, the distribution of Figure 1-b) meets all the lower and upper bounds determined by the belief function (4). Indeed: $p(x) = 1/3 \geq b(x) = 0$, $p(\{x, y\}) = 1/3 + 2/3 = 1 \geq b(\{x, y\}) = m_b(\{x, y\}) = 2/3$, etcetera.

Consistent probabilities as inner Bayesian approximations. Finally, consistent probabilities can be seen as the set of all the probabilities *more committed* than the original b.f. b.

The "least commitment principle" [6] postulates that, given a set of b.p.a.s compatible with a number of constraints, the most appropriate mass functions is the "least informative". As pointed out by Denoeux [7], in some sense it plays a role similar to that of maximum entropy in probability theory. There are many ways of measuring the information content of a belief function. This is done in practice by defining a partial order in the space of belief functions [8,9,24]. If we adopt the order relation called *weak inclusion*

$$b \leq b' \equiv b(A) \leq b'(A)\ \ \forall A \subseteq \Theta, \tag{5}$$

according to which a belief function b' dominates another b.f. b if the belief values of b' are greater than those of b for all events $A \subseteq \Theta$, the consistent probabilities (1) are exactly the group of Bayesian belief functions more committed than b according to (5). They therefore assume the meaning of *inner Bayesian approximations* of the original belief function b.

3 Vertices of Consistent Probabilities and Permutations

The region of all consistent probabilities or inner Bayesian approximations of any given belief function b is a *polytope*, i.e. the convex closure of a finite number

of probabilities in the probability simplex. Given an arbitrary belief function b with focal elements $E_1, ..., E_m$, we can define for each choice of m representatives $\{x_1, ..., x_m\}$, $x_i \in E_i$ $\forall i$, the (*extremal*) probability measure

$$b_{x_1 ... x_m} \doteq \sum_{i=1}^{m} m_b(E_i) b_{x_i}. \tag{6}$$

i.e. the Bayesian b.f. we obtain by assigning the mass of each focal element E_i to one of its elements $x_i \in E_i$. Recall what was said above about the interpretation of focal elements as mass "floating" around in a subset of Θ.

For instance, consider again the belief function (4) of Figure 1-a. If we take as representative x_1 of $E_1 = \{x, y\}$ the element y and as representative x_2 of $E_2 = \{y, z\}$ the element z, we obtain the extremal probability

$$b_{y,z} : m_{b_{y,z}}(y) = m_b(\{x, y\}) = 2/3, \qquad m_{b_{y,z}}(z) = m_b(\{y, z\}) = 1/3$$

of Figure 1-c. Note that instead the probability of Figure 1-b, even though is consistent with b, cannot be obtained this way. It is well known that $\mathcal{P}[b]$ is indeed the polytope formed by the convex closure of those extremal probabilities [23]:

$$\mathcal{P}[b] = Cl(b_{x_1 ... x_m}, \{x_1, ..., x_m\} \in E_1 \times \cdots \times E_m). \tag{7}$$

That, though, does not imply that all the points (6) are actual vertices of the simplex $\mathcal{P}[b]$, as several of them may lie on some side of the polytope, i.e., be expressed as a convex combination of the others. In fact, as we show here, the actual vertices of $\mathcal{P}[b]$ can be found in a much smaller set of probabilities, each one associated with a different permutation of the elements of Θ.

Theorem 1. *The simplex $\mathcal{P}[b]$ of the probability measures consistent with a b.f. b is the polytope $\mathcal{P}[b] = Cl(p^\rho[b]\ \forall \rho)$, where ρ is any permutation $\{x_{\rho(1)}, ..., x_{\rho(n)}\}$ of the singletons of Θ, and the vertex $p^\rho[b]$ is the Bayesian b.f. such that*

$$p^\rho[b](x_{\rho(i)}) = \sum_{A \ni x_\rho(i), A \not\ni x_\rho(j) \forall j<i} m_b(A). \tag{8}$$

Each probability function (8) attributes to each singletons $x = x_{\rho(i)}$ the mass of all focal elements of b which contains it, but does not contain the elements which precede x in the ordered list $\{x_{\rho(1)}, ..., x_{\rho(n)}\}$ generated by the permutation ρ.

In the binary case $\Theta = \{x, y\}$, for instance, it is clear that there exist only two possible permutations of singletons: $\rho^1 : \{x, y\}$, $\rho^2 : \{y, x\}$. They correspond to the following probabilities (Figure 3):

$$p^{x,y}(x) = \sum_{A \supseteq \{x\}} m_b(A) = m_b(x) + m_b(\Theta),\ p^{x,y}(y) = \sum_{A \supseteq \{y\}, A \not\supseteq \{x\}} m_b(A) = m_b(y);$$
$$p^{y,x}(y) = \sum_{A \supseteq \{y\}} m_b(A) = m_b(y) + m_b(\Theta),\ p^{y,x}(x) = \sum_{A \supseteq \{x\}, A \not\supseteq \{y\}} m_b(A) = m_b(x).$$

Proof. We need to prove that:
1. Each probability $p \in \mathcal{P}$ s.t. $p(A) \geq b(A)$ for all $A \subseteq \Theta$ can be put as a convex combination of the points (8): $p = \sum_\rho \alpha_\rho p^\rho[b]$ with $\sum_\rho \alpha_\rho = 1$, $\alpha_\rho \geq 0 \ \forall \rho$;
2. Vice-versa, each conv. comb. of the $p^\rho[b]$ meets $\sum_\rho \alpha_\rho p^\rho[b](A) \geq b(A) \ \forall A \subseteq \Theta$.

Point **2** is easily proven after we notice that each probability (8) associated with a permutation ρ of elements of Θ is indeed consistent with b, i.e. $p^\rho[b](A) \geq b(A) \ \forall A$. Whatever ρ the mass of each of the subsets B of A is attributed by (8) to some element x of A: on the other side the mass of some other events $B \not\subset A$ is also given to elements of A, so that $p^\rho[b](A) = \sum_{x \in A} p^\rho[b](x) \geq \sum_{B \subseteq A} m_b(A) = b(A)$, i.e., $p^\rho[b]$ is consistent whatever the permutation ρ. Therefore

$$\sum_\rho \alpha_\rho p^\rho[b](A) \geq \sum_\rho \alpha_\rho b(A) = b(A) \sum_\rho \alpha_\rho = b(A).$$

Concerning point **1**, we recalled in Section 2 that $b'(A) \leq b(A)$ iff m_b is the result of a redistribution of the mass $m_{b'}(A)$ of each f.e. of b' to its subsets. In the case of inner Bayesian approximations the mass of each event has to be redistributed among its elements $x \in A$:

$$m_b(A) \mapsto \alpha_x^A m_b(A) \qquad \forall x \in A, \quad \sum_{x \in A} \alpha_x^A = 1. \tag{9}$$

Therefore, for all p such that $b(A) \leq p(A) \ \forall A$ the mass $p(x)$ of each $x \in \Theta$ is

$$p(x) = \sum_{A \supseteq \{x\}} m_b(A) \alpha_x^A. \tag{10}$$

To prove (1) we then need to write (10) as a convex combination of the $p^\rho[b](x)$:

$$p(x) = \sum_\rho \alpha_\rho p^\rho[b](x) = \sum_\rho \alpha_\rho \Big(\sum_{A \ni x = x_{\rho(i)}, A \not\ni x_{\rho(j)} \forall j < i} m_b(A) \Big),$$

where i is the position of the element x according to the permutation ρ.

For all $A \supseteq \{x\}$ there exists a permutation ρ such that the elements before x in $\{x_{\rho(1)}, ..., x_{\rho(n)}\}$ fall outside A. Hence the above quantity reads as $\sum_{A \supseteq \{x\}} m_b(A) (\sum_{\rho : x_{\rho(j)} \notin A \forall j < i} \alpha_\rho)$, where again $x = x_{\rho(i)}$.

In summary we need to show that the system of equations

$$\Big\{ \alpha_x^A = \sum_{\rho : x_{\rho(j)} \notin A \forall j < i, x = x_{\rho(i)}} \alpha_\rho \qquad \forall x \in \Theta, \quad \forall A \supseteq \{x\} \tag{11}$$

has at least one solution $\{\alpha_\rho\}$ such that $\sum_\rho \alpha_\rho = 1$, $\alpha_\rho \geq 0 \ \forall \rho$.

Parenthesis: proof in the ternary case. It is useful to first illustrate the existence of a convex solution to (11) in the simple but interesting case of a ternary frame $\Theta = \{x, y, z\}$. The possible permutations of singletons in this case are six: $\rho^1 = \{x, y, z\}$, $\rho^2 = \{x, z, y\}$, $\rho^3 = \{y, x, z\}$, $\rho^4 = \{y, z, x\}$, $\rho^5 = \{z, x, y\}$,

$\rho^6 = \{z, y, x\}$. As by definition $\alpha_x^x = 1$ all equations associated with a singleton generate the normalization constraint $\alpha_{\rho^1} + \alpha_{\rho^2} + \alpha_{\rho^3} + \alpha_{\rho^4} + \alpha_{\rho^5} + \alpha_{\rho^6} = 1$.

Also, many equations in the system of equations (11)

$$\begin{cases} \alpha_x^{\{x\}} = \alpha_{\rho^1} + \alpha_{\rho^2} + \alpha_{\rho^3} + \alpha_{\rho^4} + \alpha_{\rho^5} + \alpha_{\rho^6}; \\ \alpha_x^{\{x,y\}} = \alpha_{\rho^1} + \alpha_{\rho^2} + 0 + 0 + \alpha_{\rho^5} + 0; \\ \alpha_x^{\{x,z\}} = \alpha_{\rho^1} + \alpha_{\rho^2} + \alpha_{\rho^3} + 0 + 0 + 0; \\ \alpha_x^{\Theta} = \alpha_{\rho^1} + \alpha_{\rho^2} + 0 + 0 + 0 + 0; \\ \alpha_y^{\{y\}} = \alpha_{\rho^1} + \alpha_{\rho^2} + \alpha_{\rho^3} + \alpha_{\rho^4} + \alpha_{\rho^5} + \alpha_{\rho^6}; \\ \alpha_y^{\{x,y\}} = 0 + 0 + \alpha_{\rho^3} + \alpha_{\rho^4} + 0 + \alpha_{\rho^6}; \\ \alpha_y^{\{y,z\}} = \alpha_{\rho^1} + 0 + \alpha_{\rho^3} + \alpha_{\rho^4} + 0 + 0; \\ \alpha_y^{\Theta} = 0 + 0 + \alpha_{\rho^3} + \alpha_{\rho^4} + 0 + 0; \\ \alpha_z^{\{z\}} = \alpha_{\rho^1} + \alpha_{\rho^2} + \alpha_{\rho^3} + \alpha_{\rho^4} + \alpha_{\rho^5} + \alpha_{\rho^6}; \\ \alpha_z^{\{x,z\}} = 0 + 0 + 0 + \alpha_{\rho^4} + \alpha_{\rho^5} + \alpha_{\rho^6}; \\ \alpha_z^{\{y,z\}} = 0 + \alpha_{\rho^2} + 0 + 0 + \alpha_{\rho^5} + \alpha_{\rho^6}; \\ \alpha_z^{\Theta} = 0 + 0 + 0 + 0 + \alpha_{\rho^5} + \alpha_{\rho^6}. \end{cases}$$

are actually linearly dependent, like for example equations 2 and 6 for $\alpha_x^{\{x,y\}}$ and $\alpha_y^{\{x,y\}}$. This because by definition (9): $\sum_{x \in A} \alpha_x^A = 1$. After eliminating the dependencies we get a reduced system

$$\begin{cases} \alpha_z^{\Theta} = 0 + 0 + 0 + 0 + \alpha_{\rho^5} + \alpha_{\rho^6}; \\ \alpha_x^{\{x,y\}} = \alpha_{\rho^1} + \alpha_{\rho^2} + 0 + 0 + \alpha_{\rho^5} + 0; \\ \alpha_y^{\{y,z\}} = \alpha_{\rho^1} + 0 + \alpha_{\rho^3} + \alpha_{\rho^4} + 0 + 0; \\ \alpha_x^{\{x,z\}} = \alpha_{\rho^1} + \alpha_{\rho^2} + \alpha_{\rho^3} + 0 + 0 + 0; \\ \alpha_x^{\Theta} = \alpha_{\rho^1} + \alpha_{\rho^2} + 0 + 0 + 0 + 0; \\ \alpha_x^{\{x\}} = \alpha_{\rho^1} + \alpha_{\rho^2} + \alpha_{\rho^3} + \alpha_{\rho^4} + \alpha_{\rho^5} + \alpha_{\rho^6}; \end{cases}$$

which clearly admits as solution $\alpha_{\rho^6} = \alpha_z^{\Theta}$, $\alpha_{\rho^5} = \alpha_x^{\{x,y\}}$, $\alpha_{\rho^4} = \alpha_y^{\{y,z\}}$, $\alpha_{\rho^3} = \alpha_x^{\{x,z\}}$, $\alpha_{\rho^2} = \alpha_x^{\Theta}$, $\alpha_{\rho^1} = \alpha_x^{\{x\}}$, a valid convex combination of the $p^\rho[b]$.

General solution. As like in the ternary case the normalization constraint is in fact trivially satisfied as from (11) it follows that when $A = \{x\}$, $x \in \Theta$

$$1 = \alpha_x^x = \sum_{\rho : x_{\rho(j)} \notin \{x\} \forall j < i, x = x_{\rho(i)}} \alpha_\rho = \sum_\rho \alpha_\rho$$

i.e. $\sum_\rho \alpha_\rho = 1$. Let us denote by $x_{|A|}$ any element representative of A. Due to the normalization constraint the system of equations (11) reduces to

$$\left\{ \alpha_x^A = \sum_{\rho : x_{\rho(j)} \notin A \forall j < i, x = x_{\rho(i)}} \alpha_\rho \quad \forall A \subseteq \Theta, x \neq x_{|A|}. \right. \tag{12}$$

Again $\forall A$ s.t. $|A| = 1$ we get simply the normalization constraint.

To understand the structure of (12) consider some arbitrary ordering of the elements of Θ, $x_1, ..., x_n$. If we take as representative of any event A its last element according to this ordering, we can write (12) as

$$\begin{cases} \alpha^A_{x_k} = \sum_{\rho: x_{\rho(j)} \notin A \forall j < i, x_k = x_{\rho(i)}} \alpha_\rho, & A \supseteq \{x_k\} \\ & A \not\subset \{x_1, ..., x_k\} \end{cases} \tag{13}$$

each block associated with x_k, $k = 1, ..., n-1$. The number of equations in each block k for a frame Θ of size $|\Theta| = n$ is $|\{A \subseteq \Theta : A \supseteq \{x_k\}, A \not\subset \{x_1, ..., x_k\}\}| = |\{A \subseteq \Theta : A \supseteq \{x_k\}, A \cap \{x_1, ..., x_k\}^c \neq \emptyset\}| = 2^{i-1}(2^{n-i} - 1) = 2^{n-1} - 2^{i-1}$.

Now, all the equations of each block k involve (amongst others) the α_ρ related to permutations ρ which put x_k in the first position: $x_k = x_{\rho(1)}$, as it obviously has no predecessors so that there is no $j < i$ in the subscript of the sum in (12) or (13). The number of such permutations is clearly $(n-1)!$ (the number of possible orderings of the $n-1$ successors of x_k).

But for $n > 4$ we have that $(n-1)! \geq 2^{n-1} - 2^{i-1}$: Each block has less equations than the number of permutations associated with variables α_ρ which appear in all the equations of the block. Therefore we can assign the first term of each equation of the block to one of those variables: $\alpha^A_{x_k} = \alpha_\rho$ for some ρ which puts x_k in the first position, this for all $A : A \supseteq \{x_k\}, A \not\subset \{x_1, ..., x_k\}$ (all equations in the block).

Variables associated with the remaining permutations can be set to zero. This yields a convex solution to (12), and therefore to the original system (11).

If $n = 3$ we have seen that a solution also exists. For $n = 2$ the solution is trivial. If $n = 4$ the condition $(n-1)! \geq 2^{n-1} - 2^{i-1}$ still holds for all blocks but the first one, for which the number of equations is $2^{n-1} - 2^{i-1} = 7$ while the number of variables in common is $(n-1)! = 6$. But it suffices to use the normalization constraint to replace the equation for Θ in the block x_1 (which is in excess) with an equation for Θ in the block x_3 and obtain an equivalent system of equations which meets the desired property. □

Uniqueness. We may wonder whether all the extremal points (8) generated by distinct permutations of singletons are guaranteed to be distinct. The answer is negative. Consider a belief function with b.p.a. $m_b(x) = 0.2$, $m_b(y) = 0.1$, $m_b(z) = 0.3$, $m_b(\{x, y\}) = 0.1$, $m_b(\{y, z\}) = 0.2$, $m_b(\Theta) = 0.1$ defined on a ternary frame $\Theta = \{x, y, z\}$. In this case there are six possible element permutations. Therefore by Theorem 1 $\mathcal{P}[b]$ has as vertices

$$\begin{array}{lll} \rho^1 = \{x, y, z\} : p^{\rho^1}[b](x) = .4, & p^{\rho^1}[b](y) = .3, & p^{\rho^1}[b](z) = .3; \\ \rho^2 = \{x, z, y\} : p^{\rho^2}[b](x) = .4, & p^{\rho^2}[b](y) = .1, & p^{\rho^2}[b](z) = .5; \\ \rho^3 = \{y, x, z\} : p^{\rho^3}[b](x) = .2, & p^{\rho^3}[b](y) = .5, & p^{\rho^3}[b](z) = .3; \\ \rho^4 = \{y, z, x\} : p^{\rho^4}[b](x) = .2, & p^{\rho^4}[b](y) = .5, & p^{\rho^4}[b](z) = .3; \\ \rho^5 = \{z, x, y\} : p^{\rho^5}[b](x) = .3, & p^{\rho^5}[b](y) = .1, & p^{\rho^5}[b](z) = .6; \\ \rho^6 = \{z, y, x\} : p^{\rho^6}[b](x) = .2, & p^{\rho^6}[b](y) = .2, & p^{\rho^6}[b](z) = .6; \end{array} \tag{14}$$

and we can notice that the permutations $\rho^3 = \{y, x, z\}$ and $\rho^4 = \{y, z, x\}$ yield the same function: $p^{\rho^3}[b] = p^{\rho^4}[b]$. According to the classical expression (7) of

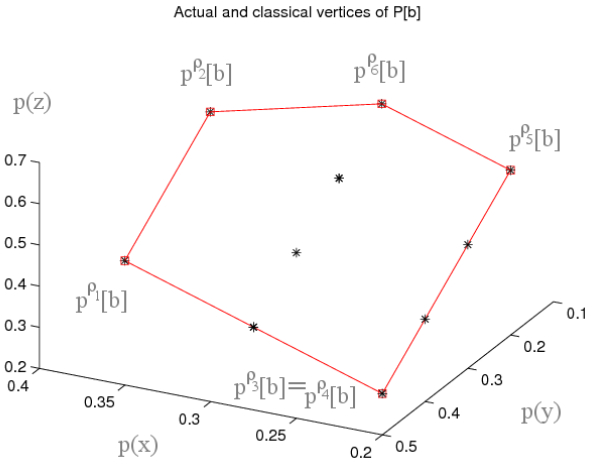


Fig. 2. The actual number of vertices (8), (14) of $\mathcal{P}[b]$ (red squares) is much smaller than the number of candidate points (6) of the classical expression. Here they are plotted as black stars for the belief function of the example. Some of them even fall inside the polytope.

$\mathcal{P}[b]$, instead, there are many more (candidate) vertices (6): $\prod_{A \subseteq \Theta : m_b(A) \neq 0} |A|$. Many fall on the sides or the interior of $\mathcal{P}[b]$ (Figure 2).

4 Bayesian and Consonant Approximations: A Symmetry

Besides including finite probabilities as a special case, belief measures also generalize finite *possibility* [25] measures, i.e. functions $Pos : 2^\Theta \to [0,1]$ on Θ such that $Pos(\bigcup_i A_i) = \sup_i Pos(A_i)$ for any family of sets $\{A_i, i \in I\}$ (where I is an arbitrary set index).

More precisely, a b.f. is *consonant* (co.b.f.) when its focal elements $\{E_i, i = 1, ..., m\}$ are nested: $E_1 \subset E_2 \subset ... \subset E_m$. As a matter of fact it can be proven that [5,26] the plausibility function pl_b associated with a belief function b on a domain Θ is a possibility measure iff b is consonant.

As possibility measures form a subclass of belief functions we can pose the problem of approximate a belief function with a possibility (or equivalently with a consonant b.f.) in perfect analogy to the case of Bayesian approximation. In particular, "outer consonant approximations" form a dual couple with inner Bayesian approximations or consistent probabilities.

Outer consonant approximations. We call *outer consonant approximations* of a belief function b [22] all the co.b.f.s which are *less committed* than the original belief function b: $\mathcal{O}[b] = \{co \in \mathcal{CO} : co(A) \leq b(A) \ \forall A \subseteq \Theta\}$. Here $\mathcal{CO}$ denotes the set of all consonant b.f.s.

According to the interpretation of the weak inclusion relation (5), $b' \leq b$ is equivalent to say that b' is obtained by letting the mass of each focal element A of b float to one or more events containing A: $B \supseteq A$.

If b' is also consonant, its focal elements have to form a chain $E_1 \subset \cdots \subset E_n$, $|E_i| = i$. An outer consonant approximation of b is then obtained by letting the mass of each f.e. be re-distributed to one or more elements of the chain.

Example. As an example, an outer consonant approximation of the belief function (4) of Figure 1 can be obtained by re-assigning the mass $2/3$ of $A = \{x, y\}$ one half $(1/3)$ to $\{x, y\}$ itself and one half $(1/3)$ to $\Theta \supset \{x, y\}$, and the mass of $A = \{y, z\}$ to $\Theta \supset \{y, z\}$ also. What we get is a consonant b.f. with focal elements $\{x, y\} \subset \Theta$ and b.p.a. $m'(\{x, y\}) = 2/3$, $m'(\Theta) = 1/3$.

Outer consonant approximations generated by permutations. In particular, with the purpose of finding outer approximations which are minimal with respect to the weak inclusion relation (5), Dubois and Prade [21] have introduced a family of outer consonant approximations obtained by considering all permutations ρ of the elements $\{x_1, ..., x_n\}$ of the frame of discernment Θ: $\{x_{\rho(1)}, ..., x_{\rho(n)}\}$. A family of nested sets can be then built

$$\left\{S_1^\rho = \{x_{\rho(1)}\}, S_2^\rho = \{x_{\rho(1)}, x_{\rho(2)}\}, \cdots, S_n^\rho = \{x_{\rho(1)}, ..., x_{\rho(n)}\}\right\} \tag{15}$$

so that a new consonant belief function co^ρ can be defined with b.p.a.

$$m_{co^\rho}(S_j^\rho) = \sum_{i: E_i \subseteq S_j^\rho, E_i \not\subset S_{j-1}^\rho} m_b(E_i). \tag{16}$$

S_j^ρ concentrates all the mass of the f.e.s E_i of b included in S_j^ρ but not in S_{j-1}^ρ.

Example. Let us consider again the belief function (4). A possible permutation of the singletons of Θ is, for instance, $\rho = \{x_{\rho(1)}, x_{\rho(2)}, x_{\rho(3)}\} = \{y, z, x\}$.
This permutation generates the following list of nested sets (15):

$$\left\{S_1^\rho = \{x_{\rho(1)}\} = \{y\}, S_2^\rho = \{x_{\rho(1)}, x_{\rho(2)}\} = \{y, z\}, \right.$$
$$\left. S_3^\rho = \{x_{\rho(1)}, x_{\rho(2)}, x_{\rho(3)}\} = \{x, y, z\}\right\}.$$

By Equation (16) we assign to $S_1^\rho = \{y\}$ the mass of all focal elements (4) of b included in $\{y\}$: there are none, so that $m_{co^\rho}(\{y\}) = 0$. To $S_2^\rho = \{y, z\}$ we assign the mass of all f.e.s inside $\{y, z\}$ not contained in $\{y\}$, i.e. the mass $1/3$ of $\{y, z\}$ itself. Finally, $S_3^\rho = \{x, y, z\}$ is assigned the mass of all f.e.s which are subsets of $\{x, y, z\}$, but not of $S_2^\rho = \{y, z\}$, namely the mass $2/3$ of $\{x, y\}$.

Binary case. In the binary case (Figure 3) a compelling symmetry emerges between $\mathcal{O}[b]$ and $\mathcal{P}[b]$. Given the definition of weak inclusion (5) it is straightforward to recognize that (as the abscissa measures the degree of belief $b(x)$ of x, and the ordinate the d.o.f. $b(y)$) the sets of inner Bayesian and outer consonant approximations form respectively a segment $\mathcal{P}[b]$ delimited by a pair of probabilities, and the union of two segments $O_{x,\Theta}[b]$ and $O_{y,\Theta}[b]$.

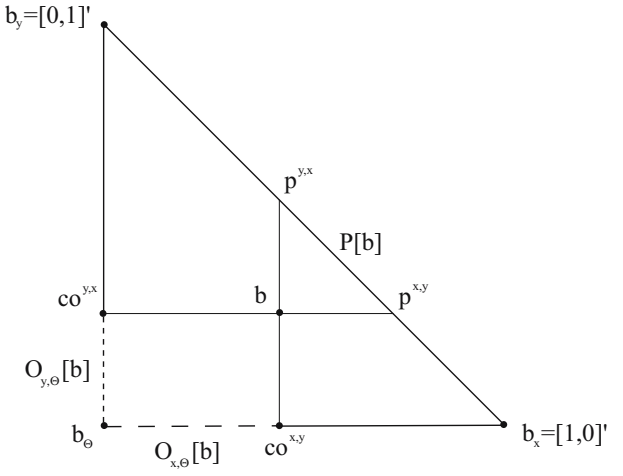


Fig. 3. Geometry of outer consonant approximations and inner Bayesian approximations in the binary case. Each b.f. is represented as a point b of coordinates $b(x), b(y)$. The set of all b.f.s on $\{x, y\}$ is the triangle in the figure. All probabilities lie on the line b_x, b_y. Inner Bayesian $P[b]$ and outer consonant $O[b]$ approximations are highlighted.

From Figure 3 we can notice the existence of an apparent bijection between vertices of $\mathcal{O}[b]$ and $\mathcal{P}[b]$. The vertex $p^{y,x}$ of the interval of consistent probabilities has the same belief value on x as the vertex $co^{x,y}$ of outer consonant approximation associated with the opposite permutation $\{x, y\}$.

In fact we can prove that the outer consonant approximations (16) generated by permutations of singletons form a subset of the vertices of $\mathcal{O}[b]$ [27] even in the general case. Furthermore, the correspondence between vertices of $\mathcal{O}[b]$ and $\mathcal{P}[b]$ generated by permutations of singletons hold in the general case.

1-1 Correspondence. Indeed, the family of outer consonant approximations (16) generated by permutations of singletons is linked by a very elegant geometric duality to the vertices (8) of the polytope of consistent probabilities $\mathcal{P}[b]$.

Let us go back to the example of the ternary frame $\Theta = \{x, y, z\}$. Given for instance the trivial permutation $\rho = \{x, y, z\}$ the vertex co^ρ has focal elements $\{x\}$, $\{x, y\}$, and $\{x, y, z\}$ and b.p.a.

$$m_{co^\rho}(\{x,y\}) = \sum_{A\subseteq\{x,y\}, A\not\subset\{x\}} m_b(A) = m_b(y) + m_b(\{x,y\}); \quad m_{co^\rho}(x) = m_b(x);$$
$$m_{co^\rho}(\Theta) = \sum_{A\subseteq\Theta, A\not\subset\{x,y\}} m_b(A) = m_b(z) + m_b(\{x,z\}) + m_b(\{y,z\}) + m_b(\Theta).$$

Consider instead the reverse permutation $\bar{\rho} = \{z, y, x\}$. The corresponding vertex $p^{\bar{\rho}}$ of $\mathcal{P}[b]$ has b.p.a.

$$p^{\bar\rho}(z) = \sum_{A \supseteq \{z\}} m_b(A) = m_b(\{z\}) + m_b(\{x,z\}) + m_b(\{y,z\}) + m_b(\Theta);$$
$$p^{\bar\rho}(y) = \sum_{A \supseteq \{y\}, A \not\supset \{z\}} m_b(A) = m_b(\{y\}) + m_b(\{x,y\});$$
$$p^{\bar\rho}(x) = \sum_{A \supseteq \{x\}, A \not\supset \{y\}, \{z\}} m_b(A) = m_b(x)$$

i.e. the b.p.a.s of co^ρ and $p^{\bar\rho}$ *coincide*. This is true in the general case.

Theorem 2. *There exists a 1-1 correspondence between the vertices co^ρ (16) of the set of outer consonant approximations of b generated by a permutation ρ of the singletons, and the vertices $p^{\bar\rho}$ (8) of the polytope of probabilities consistent with b (all of which are associated with permutations of singletons), s.t.*

$$p^{\bar\rho}(x_{\bar\rho(i)}) = m_{co^\rho}(\{x_{\rho(1)}, ..., x_{\rho(n-i+1)}\}) \tag{17}$$

i.e. their b.p.a.s on $\{S^\rho_n, ..., S^\rho_1\}$, $\{x_{\bar\rho(1)}, ..., x_{\bar\rho(n)}\}$ respectively coincide.

Proof. It suffices to show that, as $\bar\rho(i) = \rho(n-i+1)$,

$$p^{\bar\rho}(x_{\bar\rho(i)}) = \sum_{A \ni x_{\bar\rho(i)}, A \not\ni x_{\bar\rho(j)} \forall j < i} m_b(A) = \sum_{A \ni x_{\rho(n-i+1)}, A \not\ni x_{\rho(j)} \forall j > n-i+1} m_b(A)$$
$$= \sum_{\substack{A \subseteq \{x_{\rho(1)}, \cdots, x_{\rho(n-i+1)}\}, \\ A \not\subset \{x_{\rho(1)}, \cdots, x_{\rho(n-i)}\}}} m_b(A) = m_{co^\rho}(\{x_{\rho(1)}, ..., x_{\rho(n-i+1)}\}) = m_{co^\rho}(S^\rho_{n-i+1}).$$

5 Conclusions

Belief functions possess a strong credal semantics in terms of convex sets of probability distributions or consistent probabilities, for whose values on all events belief values provide lower bounds. These probabilities can also be seen as more committed or "inner" Bayesian approximations of the original b.f.

In this paper we proved a more compact form of the polytope of consistent probabilities, as the latter has $n!$ (candidate) vertices each corresponding to a different permutation of the elements of the domain. This unveils an interesting link with the vertices of the polytopes of outer consonant approximations also generated by permutations of singletons both in terms of their analytical expression and their convex geometry.

References

1. Haenni, R., Romeijn, J., Wheeler, G., Williamson, J.: Possible semantics for a common framework of probabilistic logics. In: Huynh, V.N., Nakamori, Y., Ono, H., Lawry, J., Kreinovich, V., Nguyen, H.T. (eds.) UncLog 2008, International Workshop on Interval/Probabilistic Uncertainty and Non-Classical Logics, Ishikawa, Japan. Advances in Soft Computing, vol. 46, pp. 268–279 (2008)

2. Mercier, D., Denoeux, T., Masson, M.: Refined sensor tuning in the belief function framework using contextual discounting. In: IPMU (2006)
3. Demotier, S., Schon, W., Denoeux, T.: Risk assessment based on weak information using belief functions: a case study in water treatment. IEEE Transactions on Systems, Man and Cybernetics, Part C 36(3), 382–396 (2006)
4. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)
5. Dubois, D., Prade, H.: Possibility theory. Plenum Press, New York (1988)
6. Smets, P.: Belief functions: the disjunctive rule of combination and the generalized Bayesian theorem. International Journal of Approximate Reasoning 9, 1–35 (1993)
7. Denoeux, T.: A new justification of the unnormalized Dempster's rule of combination from the Least Commitment Principle. In: Proceedings of FLAIRS 2008, Special Track on Uncertaint Reasoning (2008)
8. Yager, R.R.: The entailment principle Dempster-Shafer granules. International Journal of Intelligent Systems 1, 247–262 (1986)
9. Dubois, D., Prade, H.: A set-theoretic view of belief functions: logical operations and approximations by fuzzy sets. Int. J. of General Systems 12, 193–226 (1986)
10. Levi, I.: The enterprise of knowledge: An essay on knowledge, credal probability, and chance. The MIT Press, Cambridge (1980)
11. Chateauneuf, A., Jaffray, J.Y.: Some characterizations of lower probabilities and other monotone capacities through the use of Möbius inversion. Mathematical Social Sciences 17, 263–283 (1989)
12. Ha, V., Haddawy, P.: Geometric foundations for interval-based probabilities. In: Cohn, A.G., Schubert, L., Shapiro, S.C. (eds.) KR 1998: Principles of Knowledge Representation and Reasoning, pp. 582–593. Morgan Kaufmann, San Francisco (1998)
13. Smets, P.: The nature of the unnormalized beliefs encountered in the transferable belief model. In: Proceedings of the 8th Annual Conference on Uncertainty in Artificial Intelligence (UAI-92), San Mateo, CA, Morgan Kaufmann (1992) 292–29
14. Dubois, D., Grabisch, M., Prade, H., Smets, P.: Using the transferable belief model and a qualitative possibility theory approach on an illustrative example: the assessment of the value of a candidate. Intern. J. Intell. Systems (2001)
15. Snow, P.: The vulnerability of the transferable belief model to dutch books. Artificial Intelligence 105, 345–354 (1998)
16. Smets, P., Kennes, R.: The transferable belief model. AI 66(2), 191–234 (1994)
17. Melkonyan, T., Chambers, R.: Degree of imprecision: Geometric and algebraic approaches. International Journal of Approximate Reasoning (2006)
18. Cozman, F.G.: Calculation of posterior bounds given convex sets of prior probability measures and likelihood functions. Journal of Computational and Graphical Statistics 8(4), 824–838 (1999)
19. Seidenfeld, T., Wasserman, L.: Dilation for convex sets of probabilities. Annals of Statistics 21, 1139–1154 (1993)
20. Seidenfeld, T., Schervish, M., Kadane, J.: Coherent choice functions under uncertainty. In: Proceedings of ISIPTA 2007 (2007)
21. Baroni, P.: Extending consonant approximations to capacities. In: Proceedings of IPMU, pp. 1127–1134 (2004)
22. Dubois, D., Prade, H.: Consonant approximations of belief functions. International Journal of Approximate Reasoning 4, 419–449 (1990)
23. Cuzzolin, F.: A geometric approach to the theory of evidence. IEEE Trans. Systems, Man, and Cybernetics C 38(3) (in press, 2008)

24. Denoeux, T.: Conjunctive and disjunctive combination of belief functions induced by non distinct bodies of evidence. Artificial Intelligence (2007)
25. Dubois, D., Prade, H., Smets, P.: New semantics for quantitative possibility theory. In: ISIPTA, pp. 152–161 (2001)
26. Joslyn, C.: Towards an empirical semantics of possibility through maximum uncertainty. In: Lowen, R., Roubens, M. (eds.) Proc. IFSA 1991, pp. 86–89 (1991)
27. Cuzzolin, F.: The geometry of consonant belief functions: Simplicial complexes of possibility measures. Fuzzy Sets and Systems (under review) (2007)

A Fluent Calculus Semantics for ADL with Plan Constraints

Conrad Drescher and Michael Thielscher

Department of Computer Science,
Dresden University of Technology
Nöthnitzer Str. 46, 01187 Dresden, Germany

Abstract. Plan constraints are the most recent addition to the ever growing Planning Domain Definition Language (PDDL). In this work we consider the PDDL fragment consisting of basic ADL extended by plan constraints. We provide a purely declarative semantics for this fragment by interpreting it in the basic Fluent Calculus. We thus obtain a logical semantics for this fragment of PDDL instead of the usual meta-theoretical state transition semantics.

1 Introduction

Research in specialized planning languages originates with STRIPS [1]. Over the years this basic language has seen numerous extensions: first to the language ADL [2] and then to PDDL [3], the ever growing language that underlies the annual planning competitions. The new feature in the most recent version PDDL 3.0 [4] are plan constraints that allow to express both requirements and preferences with regard to plan quality.

Traditionally the semantics of planning languages is given in terms of state transitions. There also is a parallel line of research that seeks to provide a logical semantics for planning languages. Such complementary semantics exist for STRIPS [1,5] and ADL [6,7]. The recent works [8,9] aim at successively covering all of the semantics of PDDL 2.1 [10].

In this work we provide a purely declarative semantics for the fragment of PDDL 3.0 consisting of basic ADL and plan constraints. We do so by interpreting this fragment in the basic Fluent Calculus. The resulting system is both natural and expressive.

2 Preliminaries

In this section we recall the theoretical basis upon which our work rests. We start by recalling the basics of Fluent Calculus. Then we identify the fragment of PDDL under consideration — ADL with plan constraints.

2.1 Fluent Calculus

The Fluent Calculus [11] can be seen as a modern extension of the classical Situation Calculus [12]. One of the major differences between Fluent and Situation Calculus is that the former is action-centered while the latter is fluent-centered — at least in the

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 140–152, 2008.

popular version based on Reiter's successor state axioms [12]. That is to say, in Fluent Calculus we specify for each *action* the effects it has while a successor state axiom specifies for a *fluent* by which actions it is affected. Thus Fluent Calculus arguably is closer to current planning languages, which are also action-centered.

Our work uses a reformulation of the basic Fluent Calculus in a recently proposed unifying action calculus (UAC) [13] that allows us to keep the technical overhead to a minimum. A comprehensive treatment of the classical Fluent Calculus — which also captures the notion of state, i.e. of collections of fluents — can be found in [11].

Unifying Action Calculus. The UAC has been introduced with the stated goal of bundling research efforts in the reasoning about action community; it has been shown to encompass the Event, Fluent, and Situation Calculus, as well as planning languages such as ADL.

Formally, the UAC is based on many-sorted first order logic with equality and the four sorts TIME, FLUENT, OBJECT, and ACTION.[1] Fluents are reified, i.e. modeled as terms, and the predicate Holds : FLUENT × TIME is used to indicate whether a particular fluent evaluates to true at a particular time. For axiomatizing action preconditions the predicate Poss : ACTION × TIME × TIME is used.[2] There are only finitely many function symbols into sorts FLUENT and ACTION, respectively.

The UAC abstracts from a particular time structure. It can be instantiated, e.g., by the natural numbers that serve as the linear time structure of the Event Calculus, or by situations that provide the branching time structure of the Fluent and Situation Calculus.

Fluent Calculus Domains. Fluent Calculus domains are axiomatized in the UAC with the help of the following formula types:

Definition 1 (Basic Formulas)
For $\bar{\mathbf{s}}$*, a sequence of variables of sort* TIME*, a* state formula $\Phi[\bar{\mathbf{s}}]$ in $\bar{\mathbf{s}}$ *is a first-order formula with free variables* $\bar{\mathbf{s}}$ *and where*

- *for each occurrence of* $Holds(f, s)$ *we have* $s \in \bar{\mathbf{s}}$*;*
- *predicate Poss does not occur.*

Let A be a function into sort ACTION.

- *A* domain constraint *is a state formula in s:*

$$(\forall s)\delta[s].$$

- *A* precondition axiom *is of the form*

$$(\forall)Poss(A(\bar{\mathbf{x}}), s_1, s_2) \equiv \pi_A[s_1] \wedge s_2 = Do(A(\bar{\mathbf{x}}), s_1),$$

where $\pi_A[s_1]$ *is a state formula in* s_1 *with free variables among* $s_1, \bar{\mathbf{x}}$.[3]

[1] By convention variable symbols s, f, x, and a are used for terms of sort TIME, FLUENT, OBJECT, and ACTION, respectively.

[2] Having two arguments of sort TIME allows to model actions with duration or indirect effects.

[3] By $(\forall)\varphi$ we denote the universal closure of φ.

– *An* effect axiom *is of the form*

$$(\forall)\mathit{Poss}(A(\bar{\mathbf{x}}), s, t) \supset$$
$$\bigvee_k (\exists \bar{\mathbf{y}}_k)(\Phi_k[s] \wedge (\forall f)[\bigvee_i f = f_{ki} \vee \mathit{Holds}(f, s) \wedge \bigwedge_j f \neq g_{kj} \equiv \mathit{Holds}(f, t)])$$

The f_{ki} and g_{kj} are fluent terms with variables among $\bar{\mathbf{x}}, \bar{\mathbf{y}}_k$ and denote the positive and negative effects, respectively. The Φ_k are state formulas in s with free variables among $s, \bar{\mathbf{x}}, \bar{\mathbf{y}}$ and represent conditions under which the effects materialize. Positive and negative action effects are subject to a natural consistency assumption, namely, we require that

$$\bigwedge_i \bigwedge_j f_{ki} \neq g_{kj}$$

holds for all $k = 1, \ldots, n$.

– *An* initial state axiom *is a state formula in the least element S_0 of sort* TIME.

– *Foundational axioms Σ_{aux} contain a first order axiomatization of situations (the underlying time structure). It is based on two functions into sort* TIME*; the constant S_0 denotes the initial situation and the function Do of sort* ACTION $\times$ TIME *is used to construct successor situations:*

$$(\forall)\mathit{Do}(a_1, s_1) = \mathit{Do}(a_2, s_2) \equiv a_1 = a_2 \wedge s_1 = s_2$$
$$(\forall)\neg s \sqsubset S_0$$
$$(\forall)s \sqsubset \mathit{Do}(a, s') \equiv s \sqsubseteq s'$$
$$\phi[S_0] \wedge (\forall s, a)(\phi[s] \supset \phi[\mathit{Do}(a, s)]) \supset (\forall s')\phi[s']$$

where in the axiom scheme on the last line ϕ ranges over all state formulas in s, with only s free. Foundational axioms Σ_{aux} also contain unique name axioms for sorts ACTION *and* FLUENT*; that is, an axiom of the form*

$$(\forall \bar{\mathbf{x}} \forall \bar{\mathbf{y}}) \bigwedge_{i=1..n-1} \bigwedge_{j=i+1..n} T_i(\bar{\mathbf{x}}) \neq T_j(\bar{\mathbf{y}}) \wedge \bigwedge_{i=1..n} T_i(\bar{\mathbf{x}}) = T_i(\bar{\mathbf{y}}) \supset \bar{\mathbf{x}} = \bar{\mathbf{y}},$$

where the T_i range over all function symbols of the respective sorts. For dealing with arithmetic later on we introduce the sort NUMBER *and include an axiomatization of Presburger arithmetic.*

Definition 2 (Domain Axiomatizations). *A* domain axiomatization *Σ consists of a set Σ_{Poss} of precondition-, and a set $\Sigma_{Effects}$ of effect axioms, each containing one axiom for every function into sort* ACTION*, along with a finite set of domain constraints Σ_{dc}, a finite set of initial state axioms Σ_{Init}, and foundational axioms Σ_{aux}.*

Let us illustrate all the introduced notions by an axiomatization of the familiar blocks world domain:

Example 1 (Blocks World Axiomatization). The *precondition* of moving a block from some location x to location y is expressed by the following axiom:

$$\begin{aligned}(\forall)\text{Poss}(\text{Move}(\text{block}_1, x, y), s_1, s_2) \equiv \\ \text{Holds}(\text{On}(\text{block}_1, x), s_1) \wedge x \neq y \wedge \\ (\neg\exists\text{block}_2)\text{Holds}(\text{On}(\text{block}_2, \text{block}_1), s_1) \wedge \\ (\neg\exists\text{block}_3)(\text{Holds}(\text{On}(\text{block}_3, y), s_1) \vee y = \text{Table}) \wedge \\ s_2 = \text{Do}(\text{Move}(\text{block}_1, x, y), s_1).\end{aligned}$$

The *effects* of moving a block are axiomatized as follows:

$$\begin{aligned}&(\forall)\text{Poss}(\text{Move}(\text{block}_1, x, y), s_1, s_2) \supset \\ &\quad[(\forall f)(f = \text{On}(\text{block}_1, y) \vee (\text{Holds}(f, s_1) \wedge f \neq \text{On}(\text{block}_1, x))) \equiv \text{Holds}(f, s_2)].\end{aligned}$$

The following *domain constraint* expresses the fact that every block is situated at exactly one location[4]:

$$(\exists!y)\text{Holds}(\text{On}(x, y), s).$$

Finally, suppose that the following axiom describes what is known about the initial situation:

$$(\forall f)\text{Holds}(f, S_0) \equiv f = \text{On}(\text{Block}_1, \text{Table}) \vee f = \text{On}(\text{Block}_2, \text{Table}).$$

It can be easily verified that this axiomatization, together with the unique name axioms for the blocks and the table, entails

$$\text{Holds}(\text{On}(\text{Block}_2, \text{Block}_1), \text{Do}(\text{Move}(\text{Block}_2, \text{Table}, \text{Block}_1), S_0)).$$

3 ADL with Plan Constraints

In this section we introduce the fragment of PDDL 3.0 that we consider in this work — ADL with plan constraints. For a general introduction to action languages based on state transition semantics the reader is referred to [14].

3.1 ADL

ADL has originally been introduced to cover the expressive middle-ground between STRIPS and the Situation Calculus. It still plays an important role in the sequential, deterministic part of the international planning competitions.

Definition 3 (ADL Signature). *An ADL signature is based on a finite set of types, where types may also be defined as unions of other types. The basic type* OBJECT *is always included. The signature then contains a finite set of typed constants $\mathcal{C}$ and typed variables $\mathcal{V}$. It also includes a finite set of typed fluents $\mathcal{F}$ of arity ≥ 0 and likewise a finite set of typed operator names $\mathcal{A}$ with associated arity.*

[4] By $\exists!x\phi[x]$ we abbreviate the first order formula expressing that there is exactly one x such that $\phi[x]$.

Planning problems are expressed in ADL with the help of the following constructs:

Definition 4 (Basic ADL Constructs)

- *A state formula $\phi[\bar{\mathbf{x}}]$ is a first order formula with free variables among $\bar{\mathbf{x}}$ containing as atoms only fluents $F(\bar{\mathbf{t}})$ and equalities $\bar{\mathbf{t}}_1 = \bar{\mathbf{t}}_2$.*
- *An effect formula is the universal closure of a first order conjunction built from the following inductively defined admissible components:*
  - *fluent literals $F(\bar{\mathbf{t}})$ and $\neg F(\bar{\mathbf{t}})$ are admissible;*
  - *if ϕ and ψ are admissible then the conjunction $\phi \wedge \psi$ and the universally quantified $(\forall \bar{\mathbf{x}})\phi$ are;*
  - *if ϕ is a state formula and ψ is admissible with no occurrence of $\Rightarrow$ or $\forall$ then $\phi \Rightarrow \psi$ is.*
- *For an operator name $A \in \mathcal{A}$ the* ADL operator *A is a triple $\langle \bar{\mathbf{x}}, \pi_A, \epsilon_A \rangle$, where*
  - *the variables $\bar{\mathbf{x}}$ denote the operator's typed* parameters *(possibly zero);*
  - *the* state formula *$\pi_A[\bar{\mathbf{x}}]$ denotes the operator's precondition; and*
  - *the* effect formula *$\epsilon_A[\bar{\mathbf{x}}]$ denotes the operator's effects.*

The $\Rightarrow$ construct is not to be confused with implication; its purpose is to relate states and successor states. The definition ensures that to the right of the $\Rightarrow$ construct there is always a conjunction of fluent literals. We proceed by defining a normal form for ADL operators; the informed reader should note that this definition deviates from the one used in [9,13].

Definition 5 (Operator Normal Form). *An ADL operator A is in* normal form *if its effect formula has the following syntactic form:*

$$\bigvee_k (\forall \bar{\mathbf{x}}_k)\phi_k[\bar{\mathbf{x}}_k] \Rightarrow \delta_k[\bar{\mathbf{x}}_k]$$

where $\phi_k[\bar{\mathbf{x}}_k]$ is a state formula with free variables among $\bar{\mathbf{x}}_k$ and $\delta_k[\bar{\mathbf{x}}_k]$ is a conjunction of fluent literals with free variables among $\bar{\mathbf{x}}_k$. Further, we require that all ϕ_k are mutually exclusive.

The following proposition states that we lose nothing by making this operator normal form mandatory:

Proposition 1 (Operator Normal Form). *For every effect formula there exists an equivalent effect formula in normal form.*

Proof (Sketch). The key observation is that we can always replace e.g. $\top \Rightarrow \delta_1 \wedge \phi \Rightarrow \delta_2$ by the formula $(\phi \Rightarrow \delta_1 \wedge \delta_2) \vee (\neg\phi \Rightarrow \delta_1)$. □

Observe that rewriting an operator to normal form may introduce an exponential blowup.

Example 2 (Blocks World ADL Operator). The following is an ADL operator in normal form for the action Move(block, x, y) in the blocks world:

$$\begin{aligned} \text{Precondition: } & \text{On}(\text{block}_1, x) \wedge x \neq y \wedge \\ & (\neg\exists \text{block}_2)\text{On}(\text{block}_2, \text{block}_1) \wedge \\ & (\neg\exists \text{block}_3)(\text{On}(\text{block}_3, y) \vee y = \text{Table}) \\ \text{Effects: } & \top \Rightarrow \text{On}(\text{block}_1, x) \wedge \neg\text{On}(\text{block}_1, y) \end{aligned}$$

Definition 6 (ADL Problem Descriptions). *An ADL planning problem consists of:*

- *an ADL operator in normal form for each operator name;*
- *an initial state specification in the form of a conjunction of ground fluent literals; and*
- *a goal description in the form of a closed state formula.*

The following is an ADL problem description analogous to the Fluent Calculus domain from example 1:

Example 3 (ADL Blocks World Description). The only operator shall be as given in example 2 above. Let the initial state be specified by $\mathrm{On}(\mathrm{Block}_1, \mathrm{Table}) \wedge \mathrm{On}(\mathrm{Block}_1, \mathrm{Table})$ and the goal consist of stacking up all blocks, axiomatized as $(\exists!\mathrm{block})\mathrm{On}(\mathrm{block}, \mathrm{Table})$.

ADL admits both open and closed world reasoning – in the open world case the truth-value of fluent literals may be unknown. The existing state transition semantics for ADL from [10], however, is based on the closed world assumption. In this setting the initial state specification is a conjunction of ground fluent atoms. This specification is completed by adding the negation of every ground fluent atom that does not yet occur in the initial state specification, so that eventually every ground fluent atom of the language or its negation occurs in the initial state specification.

The semantics of ADL also makes strong assumptions about the meaning of the constants $\mathcal{C}$: no two constants denote the same object (uniqueness of names) and all existing objects are named by some constant. This latter requirement allows for substitutional quantification: e.g. a subformula $(\forall x)P(x)$ can equivalently be written as $\bigwedge_{C_i} P(C_i)$ where the C_i are all the constants of the domain. Thus, although ADL uses the language of first order logic it does not employ first order semantics.

A plan for an ADL planning problem is a ground sequence $\langle A_1(\bar{\mathbf{t}}_1), \ldots, A_n(\bar{\mathbf{t}}_n)\rangle$ of operators A_i with constants $\bar{\mathbf{t}}_i$ substituted for the parameters $\bar{\mathbf{x}}_i$. A plan is a solution for the planning problem iff the state obtained by sequentially applying the operators $A_i(\bar{\mathbf{t}}_i)$ to the initial state yields a state satisfying the goal description.

3.2 Plan Constraints

Plan constraints allow to express both hard and soft constraints on the computed plans: "hard" means that a constraint *has* to be satisfied while "soft" means that it *should*, if possible.

State Trajectory Constraints. State trajectory constraints are the hard constraints. They allow to express that some property has to hold throughout/at some point/etc. in the plan.

Formally, state trajectory constraints are handled by introducing modalities that can be used in goal descriptions. The available modalities are `at end`, `always`, `sometime`, `at-most-once`, `sometime-after`, and `sometime-before`. We omit the modalities `within` and `always-within` since these require an explicit notion of time that is not supported by the ADL subset of PDDL. The modalities may

not be nested. They can be combined by logical conjunction and be universally quantified from the outside. Universally quantified constraints only serve as shorthand for the equivalent ground formula.[5] For example we can enforce that a property ϕ holds throughout a plan that achieves the goal ψ by writing $\psi \wedge \texttt{always}\ \phi$.

The original semantics for state trajectory constraints in PDDL has been defined in terms of sequences of state-timepoint pairs $\langle (S_0, 0), (S_1, t_1), \ldots, (S_n, t_n) \rangle$, where the S_i denote all the states that occur during plan execution in chronological order. In the case of ADL this can be simplified to sequences of states $\langle S_0, \ldots, S_n \rangle$.

Definition 7 (Semantics of Temporal Modalities). *The semantics of the temporal modalities is then as follows:*

$$
\begin{array}{lll}
\langle S_0, \ldots, S_n \rangle \vDash \phi & \textit{iff} & S_n \vDash \phi \\
\langle S_0, \ldots, S_n \rangle \vDash \texttt{at end}\ \phi & \textit{iff} & S_n \vDash \phi \\
\langle S_0, \ldots, S_n \rangle \vDash \texttt{always}\ \phi & \textit{iff} & \forall i : 0 \leq i \leq n : S_i \vDash \phi \\
\langle S_0, \ldots, S_n \rangle \vDash \texttt{sometime}\ \phi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \vDash \phi \\
\langle S_0, \ldots, S_n \rangle \vDash \texttt{at-most-once}\ \phi & \textit{iff} & \forall i : 0 \leq i \leq n : \textit{if}\ S_i \vDash \phi\ \textit{then} \\
 & & \neg \exists j, k : i < j < k \leq n : \\
 & & S_j \vDash \neg\phi\ \textit{and}\ S_k \vDash \phi \\
\langle S_0, \ldots, S_n \rangle \vDash \texttt{sometime-after}\ \phi\ \psi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \vDash \phi\ \textit{implies} \\
 & & \exists j : i < j \leq n : S_j \vDash \psi \\
\langle S_0, \ldots, S_n \rangle \vDash \texttt{sometime-before}\ \phi\ \psi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \vDash \phi\ \textit{implies} \\
 & & \exists j : 0 \leq j < i : S_j \vDash \psi
\end{array}
$$

The expression `at-most-once` ϕ prohibits that ϕ changes its truth-value to false and back to true in the course of a plan. A constraint using the `always` modality might conflict with the initial state specification; we tacitly assume that initial state specifications ϕ do not violate any constraints. State trajectory constraints can appear both in the planning problem file and in the action domain file [4].

Preferences. Preferences are the soft constraints, i.e. properties that are desired but not required to hold. Instead of an elaborate qualitative model of preferences PDDL adopts a quantitative model.

The syntax for ADL preferences is

$$\texttt{preference}\ \phi,$$

where ϕ denotes a state formula possibly containing state trajectory constraints.[6] As in the case of state trajectory constraints preferences may not be nested, and only be combined by logical conjunction. Again, formulas of the form

$$(\forall \bar{\mathbf{x}})\texttt{preference}\ \phi(\bar{\mathbf{x}})$$

may be used as shorthand for the logical equivalent conjunction

$$\bigwedge_{\bar{\mathbf{t}}} \texttt{preference}\ \phi(\bar{\mathbf{t}})$$

[5] Recall that ADL admits substitutional quantification.

[6] But note that state trajectory constraints may not contain preferences.

where $\bar{\mathbf{t}}$ denotes all possible ground substitutions for the variables $\bar{\mathbf{x}}$. ADL preferences may occur in goals and in operator preconditions. In the latter case they must not contain the state trajectory modalities.

The semantics of preferences is simple and intuitive. A preference simply always evaluates to true. However, a preference can be satisfied or violated. Let $\langle S_0, \ldots, S_n \rangle$ denote the sequence of states corresponding to a plan. If the precondition of the operator applied to state S_i contains a preference `preference` ϕ, then the preference is violated if $\langle S_i \rangle \models \neg\phi$. Likewise a preference occurring in the planning goal is violated if $\langle S_0, \ldots, S_n \rangle \models \neg\phi$. An overall penalty is assigned to the plan and equals the sum of

- the number of `preference` ϕ expressions from operator preconditions that have been violated; and
- the number of `preference` ϕ expressions from the goal description that have been violated.

The optimal plan in the setting of ADL with plan constraints is the plan with the minimal number of preferences violated. It is worth pointing out that the notion of optimality crucially depends on complete information — in the case of open world ADL it may be impossible to identify whether a plan is optimal.

4 The Fluent Calculus Semantics for ADL with Plan Constraints

The Fluent Calculus semantics for ADL with plan constraints is obtained by correctly embedding the latter into the former.

4.1 Scope of the ADL Constraints

First off we have to decide whether the constraints from the action domain file and the planning problem file should be treated alike or not. Quoting from [4], "constraints (...) specified in the action domain file (...) might be seen as safety conditions (...) that must alway be respected in any valid plan for the domain (...)." This seems to suggest that these constraints are intended to serve a purpose similar to that of domain constraints in the Fluent Calculus. However, quoting from [15] constraints from the planning problem file "(...) are added to those (if any) in the domain file and together they represent a collection of goals that must be satisfied by any valid plan.". So we adopt the viewpoint that the constraints apply only to the planning goal and not to the action domain as a whole. Let us illustrate the issue at hand by a small example.

Example 4 (Scope of ADL Constraints). Assume that in the ADL blocks world domain from example 3 the constraint `always` $\mathrm{On}(\mathrm{Block}_1, \mathrm{Table})$ is part of the action domain file. Let Σ denote the Fluent Calculus axiomatization of this domain from example 1. If we extend Σ by the domain constraint $(\forall s)\mathrm{Holds}(\mathrm{On}(\mathrm{Block}_1, \mathrm{Table}), s)$ the resulting theory is inconsistent. Instead we extend the reasoning problem $\Sigma \models (\exists s)\phi(s)$ to the additionally qualified $\Sigma \models (\exists s)\phi(s) \wedge \neg(\exists s')s' \sqsubseteq s \wedge \neg\mathrm{Holds}(\mathrm{On}(\mathrm{Block}_1, \mathrm{Table}), s')$ — where ϕ denotes the goal description.

4.2 The Mapping

We are finally ready to define the mapping:

A Corresponding Language. We start by defining a Fluent Calculus signature based on the ADL signature. In order to simplify the presentation we will assume that ADL problems only include the single type OBJECT. Our results can easily be reformulated in an appropriately sorted version of the Fluent Calculus, so that this is without loss of generality. Given an ADL problem based on a signature with constants $\mathcal{C}$, operator names $\mathcal{A}$, and fluent predicates $\mathcal{F}$ we create a Fluent Calculus signature with corresponding constants $\mathcal{C}'$, functions into sort ACTION $\mathcal{A}'$, and functions into sort FLUENT $\mathcal{F}'$. In order to deal with preferences we introduce the additional sort NUMBER and extend the foundational axioms Σ_{aux} by an axiomatization of the natural numbers.

Based on this signature we include unique-name-axioms for sort OBJECT. Likewise we include a domain closure axiom for sort OBJECT, that is an axiom of the form:

$$(\forall x) \bigvee_{i=1..n} x = c_i,$$

where x is a variable of sort OBJECT and the c_i denote all object constants of the signature. For dealing with preferences we introduce a special fluent, Penalty$/1$, that takes a natural number as argument.

The Initial State. Mapping ADL initial state specifications ϕ to Fluent Calculus initial state axioms $\phi'[S_0]$ is done in the obvious way: replace every fluent $F(\bar{\mathbf{x}})$ in ϕ by $\text{Holds}(F'(\bar{\mathbf{x}}), S_0)$. Below for an ADL state formula ψ by $\psi'[s]$ we will denote the corresponding Fluent Calculus state formula obtained in this fashion for an arbitrary situation s. Finally we include $\text{Holds}(\text{Penalty}(0), S_0)$ into $\phi[S_0]$. The purpose of this fluent will be to accumulate the number of preferences violated.

The Operators. Mapping ADL operators to Fluent Calculus consists of creating corresponding precondition and effect axioms. First we introduce a bit of notation. Let A be an operator with operator precondition π_A. Without loss of generality we assume that π_A is of the form $\pi_{A_1} \wedge \pi_{A_2}$ where π_{A_1} is an ordinary ADL precondition and π_{A_2} is a conjunction of preferences. Then denote

- by $\Pi_{A_{\text{pref}}}$ the set consisting of the logical parts ψ_i of the preferences `preference` ψ_i from π_{A_2}; and
- by $\Pi_{A_{\text{pref-cases}}}$ the set of pairs $\langle \bigwedge_i (\neg)\psi_i, n_j \rangle$ where each $\psi_i \in \Pi_{A_{\text{pref}}}$ and $n_j \in \mathbb{N}$ equals the number of $\neg\psi_i$ that occur in $\bigwedge_i (\neg)\psi_i$ where $\psi_i \in \Pi_{A_{\text{pref}}}$.

That is to say, n_j denotes the number of preferences that are violated if $\bigwedge_i (\neg)\psi_i$ holds. There are 2^i such pairs in $\Pi_{A_{\text{pref-cases}}}$ and by construction these pairs are mutually exclusive.

For every ADL operator $A = \langle \bar{\mathbf{x}}, \pi_A, \epsilon_A \rangle$ from the planning problem we define the action precondition axiom

$$(\forall \bar{\mathbf{x}}, s)\text{Poss}(A(\bar{\mathbf{x}}), s_1, s_2) \equiv \pi'_{A_1} \wedge s_2 = \text{Do}(A(\bar{\mathbf{x}}), s).$$

Recall that we assume ϵ_A to be of the form $\bigvee_k (\forall \bar{\mathbf{x}}_k)\phi[\bar{\mathbf{x}}_k] \Rightarrow \delta[\bar{\mathbf{x}}_k]$. By Δ^+ (Δ^-) denote the set of positive (negative) literals from $\delta[\bar{\mathbf{x}}_k]$. Define the corresponding Fluent Calculus effect axiom as:

$$
\begin{aligned}
&(\forall)\text{Poss}(A(\bar{\mathbf{x}}), s_1, s_2) \supset \\
&\quad \bigvee_{k_i} \phi'[\bar{\mathbf{x}}_k, s_1] \wedge \gamma_i'[s_1] \wedge \text{Holds}(\text{Penalty}(n_1), s_1) \wedge n_2 = n_1 + n_i \wedge \\
&\qquad [\![(\forall f)[f = \text{Penalty}(n_2) \vee \bigvee_{F(\bar{\mathbf{x}}) \in \Delta^+} f = F(\bar{\mathbf{y}})] \vee \\
&\qquad\quad [\text{Holds}(f, s_1) \wedge f \neq \text{Penalty}(n_1) \wedge \bigwedge_{F(\bar{\mathbf{x}}) \in \Delta^-} f \neq F(\bar{\mathbf{x}})] \\
&\qquad \equiv \text{Holds}(f, s_2)]\!],
\end{aligned}
$$

where $\langle \gamma_i, n_i \rangle \in \Pi_{A_{\text{pref-cases}}}$. Each of the k_i disjuncts states that, if prior to action application

- the accumulated penalty equates n_1; and
- case k of the ADL operator applies,

then after action application a fluent f holds if-and-only if

- f is equal to Penalty$(n2)$ where $n2$ is the new accumulated penalty; or
- f is a positive effect of the ADL operator; or
- f does not equal Penalty$(n1)$; or
- f held prior to action application and is not a negative effect of the ADL operator.

Let us stress that all the k_i disjuncts are mutually exclusive. This completes the definition of a Fluent Calculus domain Σ corresponding to an ADL planning problem.

The Goal Descriptions. We now turn to goal descriptions. These will be mapped to Fluent Calculus queries that will be evaluated with regard to the domain axiomatization Σ. If the ADL goal description ϕ does not contain any constraints our task is easy: we simply ask whether

$$
\begin{aligned}
\Sigma \vDash\ & (\exists s, n)\phi'[s] \wedge \text{Holds}(\text{Penalty}(n), s) \wedge \\
& (\neg\exists s', n')\phi'[s'] \wedge \text{Holds}(\text{Penalty}(n'), s') \wedge n' < n.
\end{aligned} \tag{1}
$$

We proceed by extending this mapping to goal descriptions containing constraints. Without loss of generality we can assume that the goal description ϕ is of the form $\phi_1 \wedge \phi_2 \wedge \phi_3$, where

- ϕ_1 is an ordinary ADL goal;
- ϕ_2 is a conjunction of state trajectory constraints; and
- ϕ_3 is a conjunction of preferences.

The corresponding Fluent Calculus query is of the form

$$(\exists s, n, n_{\text{final}})\psi_1[s, n] \wedge \psi_2[s] \wedge \psi_3[s, n, n_{\text{final}}],$$

where $\psi_1[s, n]$

– is the formula from (1) if there are no preferences ϕ_3 in the goal description;
– is $(\exists s, n)\phi'[s] \wedge \text{Holds}(\text{Penalty}(n), s)$ otherwise.

The mapping from the hard state trajectory constraints ϕ_2 to $\psi_2[s]$ can be obtained from the base cases depicted in figure 1.

ADL constraint	Fluent Calculus subquery
`at end` ψ	$\psi'[s]$
`always` ψ	$(\forall s_1)s_1 \sqsubseteq s \supset \psi'[s_1]$
`sometime` ψ	$(\exists s_1)s_1 \sqsubseteq s \wedge \psi'[s_1]$
`at-most-once` ψ	$(\exists s_1)s_1 \sqsubseteq s \wedge \psi'[s_1] \supset$ $\neg(\exists s_2, s_3)s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s_3 \wedge s_3 \sqsubseteq s \wedge \neg\psi'[s_2] \wedge \psi'[s_3]$
`sometime-after` ψ_1 ψ_2	$(\exists s_1)s_1 \sqsubseteq s \wedge \psi_1'[s_1] \supset (\exists s_2)s_1 \sqsubseteq s_2 \wedge s_2 \sqsubseteq s \wedge \psi_2'[s_2]$
`sometime-before` ψ_1 ψ_2	$(\exists s_1)s_1 \sqsubseteq s \wedge \psi_1'[s_1] \supset (\exists s_2)s_2 \sqsubseteq s_1 \wedge \psi_2'[s_2]$

Fig. 1. Mapping State Trajectory Constraints to Fluent Calculus

For the mapping from the preferences $\phi_3 = \bigwedge_k \texttt{preference}\ \varphi_k$ in the goal description to the Fluent Calculus subquery $\psi_3[s, n, n_{\text{final}}]$ we introduce again some notation: denote by Φ_{cases} the set of pairs $\langle \bigwedge_k (\neg)\varphi_k, n_k \rangle$ where $n_k \in \mathbb{N}$ equals the number of $\neg\varphi_k$ that occur in $\bigwedge_k (\neg)\varphi_k$. Without loss of generality we assume that φ_k is of the form $\varphi_{k_1} \wedge \varphi_{k_2}$, where φ_{k_1} is an ADL state formula and φ_{k_2} is a conjunction of state trajectory constraints. We map φ_{k_1} to $\varphi'_{k_1}[s]$ and φ_{k_2} to $\varphi^*_{k_2}[s]$ — where $\varphi^*_{k_2}[s]$ is obtained analogously to the mapping from ϕ_2 to $\psi_2[s]$. With a little abuse of notation we denote $\varphi'_{k_1}[s] \wedge \varphi^*_{k_2}[s]$ by $\varphi'_k[s]$.

We now define the Fluent Calculus subquery $\psi_3[s, n, n']$ corresponding to ϕ_3 to be

$$\bigwedge_i \bigwedge_k (\neg)\varphi'_k[s] \supset n_{\text{final}} = n + n_i) \wedge$$
$$(\neg\exists s', n', n'_{\text{final}})\psi_1[s', n'] \wedge \psi_2[s'] \wedge$$
$$\bigwedge_i \bigwedge_k (\neg)\varphi'_k[s'] \supset n'_{\text{final}} = n' + n_i) \wedge n'_{\text{final}} < n_{\text{final}},$$

where $\langle \bigwedge_k (\neg)\varphi_k, n_i \rangle \in \Phi_{\text{cases}}$. This subquery ensures plan optimality by requiring that

– n_{final} is the sum of
  • The penalty n that stems from violated preferences in action preconditions and
  • The number n_i of preferences φ_k from the goal description violated by s; and
– there does not exists a situation s' satisfying the goal description ψ_1 and the hard plan constraints ψ_2 with a smaller final penalty.

4.3 Correctness of the Translation

We have defined a mapping from ADL planning problems with plan constraints to Fluent Calculus domain axiomatizations Σ and Fluent Calculus queries $(\exists s)\phi[s]$. We are now ready to state our main result:

Theorem 1 (Correctness of the Translation). *Let the Fluent Calculus domain Σ and query $(\exists s)\phi[s]$ be obtained from an ADL planning problem via our mapping. A sequence $\langle A_1(\bar{\mathbf{t}}_1), \ldots, A_n(\bar{\mathbf{t}}_n)\rangle$ of ground ADL operators $A_i(\bar{\mathbf{t}}_i)$ is an optimal solution for the planning problem if and only if $\Sigma \models \phi[Do(A_n(\bar{\mathbf{t}}_n), Do(A_{n-1}(\bar{\mathbf{t}}_{n-1}), \ldots S_0)\ldots)]$.*

Proof (Sketch). The full proof of this theorem is quite tedious and therefore omitted. However, in order to provide some evidence for the correctness of the theorem, we point out that our embedding is very generic, in the sense that the Fluent Calculus domain axiomatization Σ and the ADL planning problem correspond axiom-by-axiom. □

5 Summary

5.1 Related Work

The series of works [7,8,9] successively provided logical semantics for more and more expressive fragments of PDDL by interpreting these in a recently proposed first order modal variant of the Situation Calculus [16]. None of these works covers plan constraints yet. Many ontological features of PDDL like e.g. concurrent actions and actions with duration are not present in the basic Situation Calculus, however. Thus, in order to obtain a mapping from PDDL fragments to Situation Calculus the underlying logic had to be considerably extended. For Fluent Calculus such extensions have first been introduced in [17]. Most likely these ideas can be adapted in order to obtain Fluent Calculus semantics for equally expressive fragments of PDDL as those covered in [7,8,9].

Using a mapping defined in [13] we can obtain Situation Calculus axiomatizations corresponding to their Fluent Calculus counterparts. This immediately yields a Situation Calculus semantics for ADL with plan constraints.

Instead of plan constraints costs associated to actions have been introduced for the sequential deterministic part of this year's planning competition at ICAPS-08.[7] This system can also very naturally be interpreted in the Fluent Calculus; plan costs can be computed by summing over situation terms.

Our result identifies a fragment of the Fluent Calculus for which reasoning can be based on efficient specialized planning software instead of the more general constraint logic programming implementation Flux [11]. A tight integration may be achieved by adopting ideas from [18], where planning problems have efficiently been encoded and solved in CLP(FD). Note that reversing our mapping does not introduce an additional blowup as opposed to compiling operators to normal form.

5.2 Conclusion

We have given a purely declarative semantics for ADL with plan constraints by interpreting it in the basic Fluent Calculus. Our semantics is logical, as opposed to the only previously available semantics, which was based on state transitions. Along the way we have clarified the role played by state trajectory constraints by determining their scope. The resulting system is expressive and — since both PDDL and the Fluent Calculus are action-centered formalisms — very natural.

[7] See `http://ipc.informatik.uni-freiburg.de/`.

References

1. Fikes, R.E., Nilsson, N.J.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence 2, 189–208 (1971)
2. Pednault, E.P.D.: ADL: Exploring the Middle Ground between STRIPS and the Situation Calculus. In: Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR 1989), San Mateo, California, pp. 324–332. Morgan Kaufmann, San Francisco (1989)
3. Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL—The Planning Domain Definition Language (1998), ftp://ftp.cs.yale.edu/pub/mcdermott/software/pddl.tar.gz
4. Gerevini, A., Long, D.: Preferences and Soft Constraints in PDDL 3.0. In: Proceedings of the ICAPS-2006 Workshop on Preferences and Soft Constraints in Planning, Lake District of the UK, pp. 46–53 (2006)
5. Lifschitz, V.: On the Semantics of STRIPS. In: Reasoning about Actions and Plans, Temberline, Oregon, pp. 1–9. Morgan Kaufmann, San Francisco (1986)
6. Pednault, E.P.D.: ADL and the State-Transition Model of Action. Journal of Logic and Computation 4, 467–512 (1994)
7. Claßen, J., Lakemeyer, G.: A Semantics for ADL as Progression in the Situation Calculus. In: Proceedings of the 11th International Workshop on Non-Monotonic Reasoning (NMR 2006), Lake District, UK (2006)
8. Claßen, J., Hu, Y., Lakemeyer, G.: A Situation-Calculus Semantics for an Expressive Fragment of PDDL. In: Proceedings of the Twenty-second National Conference on Artificial Intelligence (AAAI 2007), Menlo Park, CA, pp. 956–961 (2007)
9. Claßen, J., Eyerich, P., Lakemeyer, G., Nebel, B.: Towards an Integration of Golog and Planning. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, pp. 1846–1851 (2007)
10. Fox, M., Long, D.: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research 20, 61–124 (2003)
11. Thielscher, M.: Reasoning Robots: The Art and Science of Programming Robotic Agents. Springer, Dordrecht (2005)
12. Reiter, R.: Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems. MIT Press, Cambridge (2001)
13. Thielscher, M.: A Unifying Action Calculus. Artificial Intelligence (submitted, 2007), www.fluxagent.org/publications.htm
14. Gelfond, M., Lifschitz, V.: Action Languages. Electronic Transactions on Artificial Intelligence 3 (1998), http://www.ep.liu.se/ea/cis/1998/016/
15. Gerevini, A., Long, D.: BNF Description of PDDL3.0 (2005), http://zeus.ing.unibs.it/ipc-5/
16. Lakemeyer, G., Levesque, H.J.: Situations, Si! Situation Terms, No! In: Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR 2004), Whistler, Canada, pp. 516–526 (2004)
17. Thielscher, M.: The Concurrent, Continuous Fluent Calculus. Studia Logica 67, 315–331 (2001)
18. Dovier, A., Formisano, A., Pontelli, E.: Multivalued Action Languages with Constraints in CLP(FD). In: Proceedings of the Twenty-third International Conference on Logic Programming (ICLP 2007), Porto, Portugal, pp. 255–270 (2007)

Computational Complexity of Semi-stable Semantics in Abstract Argumentation Frameworks

Paul E. Dunne[1] and Martin Caminada[2]

[1] Department of Computer Science, The University of Liverpool, UK
ped@csc.liv.ac.uk
[2] Computer Science and Communications Research Unit, University of Luxembourg
martin.caminada@uni.lu

Abstract. Semi-stable semantics offer a further extension based formalism by which the concept of "collection of justified arguments" in abstract argumentation frameworks may be described. In contrast to the better known stable semantics, one advantage of semi-stability is that any finite argumentation framework always has at least one semi-stable extension. Although there has been some development of the formal logical theory of semi-stable semantics so that several computational properties of these extensions have been identified, with the exception of some algorithmic studies, more detailed investigation of computational complexity issues has been neglected. Our purpose in this article is to present a number of results on the complexity of some natural decision questions for semi-stable semantics.

1 Introduction

The semi-stable semantics for formal argumentation can be traced back to the work of Bart Verheij [1,2] and has recently been revived as a research topic by Caminada [3]. Semi-stable semantics can be positioned between stable semantics and preferred semantics in the sense that each stable extension is also a semi-stable extension and each semi-stable extension is also a preferred extension. Moreover, for argumentation frameworks that have at least one stable extension, the set of semi-stable extensions is exactly the same as the set of stable extensions. That is, in situations where there exists at least one stable extension, stable semantics and semi-stable semantics coincide. The advantage of semi-stable semantics, however, is that for finite argumentation frameworks semi-stable extensions are always guaranteed to exist, even in situations where stable extensions do not. These properties, as well as those described in [4], make semi-stable semantics an interesting alternative to stable semantics, which, despite the criticisms raised by researchers such as Dung [5], continue to be widely used in fields like logic programming [6] and answer set programming [7].

The virtues of semi-stable semantics and how it relates to stable semantics can perhaps be illustrated by examining the virtues of paraconsistent logic and

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 153–165, 2008.

how it relates to classical logic. One of the properties of classical logic is that inconsistency can lead to a collapse of all entailment. That is, relatively small and local problems in the knowledge base can cause a global collapse of entailment. Paraconsistent logic, such as [8], has been proposed as an approach to deal with this problem. The idea of paraconsistent logic is not only to be "crash resistant" [4], meaning that no collapse can occur, but it should ideally also be "backward compatible" [4] with classical logic, meaning that in cases where there is no collapse of classical logic (no inconsistencies) the new paraconsistent formalism should yield the same outcome as classical logic.

For stable semantics, as well as for formalisms that apply stable semantics such as Default Logic [9] and Answer Set Programming [7], one can identify the same kind of issue: relatively small and local problems (such as a rule $p \leftarrow \texttt{not}\ p$) can cause a global collapse of all entailment (the absence of stable extensions). It would be desirable to have an alternative approach that is "crash resistant" [4], implying that there should always be at least one extension, but also "backward compatible" [4] to the original stable semantics. That is, in cases where stable extensions do exist, the same outcome is yielded as under stable semantics. Semi-stable semantics is an approach that satisfies these properties. One can say that semi-stable semantics relates to stable semantics in the same way as paraconsistent logic (like [8]) relates to classical logic.

Several properties of semi-stable semantics, also in relation to other semantics, have been studied by Baroni and Giacomin [10,11]. Further work on proof procedures has recently been started by Caminada, who provides an algorithm for computing the set of all semi-stable extensions, given an argumentation framework [12]. What has been missing, however, is a complexity analysis of the various decision problems associated with semi-stable semantics, especially since the the issue of computational complexity has received quite some attention with respect to other semantics in Dung's framework, e.g. the work of Dimopoulos and Torres [13] concerning basic questions in preferred and stable semantics, Dunne and Bench-Capon [14] with regard to deciding when preferred and stable semantics coincide, and the more recent developments presented by Dunne [15] dealing with complexity-theoretic questions within the ideal semantics of Dung *et al.* [16,17]. Thus, while semi-stable semantics may offer several advantages compared to other semantics (especially compared to stable [4]), it is unclear at what cost (if any) such advantages are gained. It is this question that the current paper intends to address.

The remainder of this paper is structured as follows. In Section 2 we describe the basic components of Dung's abstract model of argument from [5] together with a brief review of some notions from computational complexity theory. We then, in Section 3, describe a number of general decision questions relating to extension-based semantics in argumentation frameworks. Our results on the complexity of these questions for the specific instantiation of semi-stable semantics are given in Section 4. Finally, conclusions and open questions are reviewed in Section 5.

2 Background

The model of abstract argumentation which forms the basis for the results of this paper was proposed by Dung [5], wherein the following concepts were introduced.

Definition 1. *An* argumentation framework *(*AF*) is a pair* $\mathcal{H} = \langle \mathcal{X}, \mathcal{A} \rangle$*, in which* $\mathcal{X}$ *is a finite set of* arguments *and* $\mathcal{A} \subseteq \mathcal{X} \times \mathcal{X}$ *is the* attack relationship *for* $\mathcal{H}$*. A pair* $\langle x, y \rangle \in \mathcal{A}$ *is referred to as '*y is attacked by x*' or '*x attacks y*'. For* R*,* S *subsets of arguments in the* AF $\mathcal{H}(\mathcal{X}, \mathcal{A})$*, we say that* $s \in S$ *is* attacked *by* R *– written* $attacks(R, s)$ *– if there is some* $r \in R$ *such that* $\langle r, s \rangle \in \mathcal{A}$*. For subsets* R *and* S *of* $\mathcal{X}$ *we write* $attacks(R, S)$ *if there is some* $s \in S$ *for which* $attacks(R, s)$ *holds;* $x \in \mathcal{X}$ *is* acceptable with respect to S *if for every* $y \in \mathcal{X}$ *that attacks* x *there is some* $z \in S$ *that attacks* y*. A subset,* S*, is* conflict-free *if no argument in* S *is attacked by any other argument in* S*. For* $S \subseteq \mathcal{X}$*,*

$$\begin{array}{rcl} S^- & =_{\text{def}} & \{\ p\ :\ \exists\ q \in S\ \ \textit{such that}\ \langle p, q \rangle \in \mathcal{A}\} \\ S^+ & =_{\text{def}} & \{\ p\ :\ \exists\ q \in S\ \ \textit{such that}\ \langle q, p \rangle \in \mathcal{A}\} \end{array}$$

Conflict-free sets underpin the concept of "collection of justified arguments" advanced through a number of forms of so-called *extension-based* models. Thus, although conflict-free sets are internally consistent since no attacks are present between members, in themselves these are too weak in order to capture the idea of set of justified arguments: such sets may still be attacked by external arguments. Informally an *extension-based* semantics prescribes additional conditions that a conflict-free set of arguments ought to satisfy in order to be regarded as justifiable, hence an extension basis, $\mathcal{E}$, defines a mapping from AFs $\mathcal{H}(\mathcal{X}, \mathcal{A})$ to subsets of $\mathcal{X}$. The following models have been proposed and widely studied in previous work.

Definition 2. *In the description below,* $\mathcal{H}(\mathcal{X}, \mathcal{A})$ *is an arbitrary* AF *and* S*,* T *denote subsets of* $\mathcal{X}$*. For a given extension model,* $\mathcal{E}$*,* $\mathcal{E}(\mathcal{H})$ *are the subsets of* $\mathcal{X}$ *satisfying the conditions prescribed by* $\mathcal{E}$*.*

a. Grounded *(*GND*)*
Let $\mathcal{F}(S) = \{\ x \in \mathcal{X}\ :\ x$ *is acceptable to* $S\}$*. The* grounded extension *(*GND$(\mathcal{H})$*) is the smallest fixed point of* $\mathcal{F}$*. It is shown in [5] that* $\mathcal{F}(\emptyset) \subseteq \mathcal{F}(\mathcal{F}(\emptyset)) \subseteq \cdots \subseteq \mathcal{F}^k(\emptyset)$ *and that in (finitary)* AFs *the grounded extension is equal to* $\bigcup_{k=0}^{\infty} \mathcal{F}^k(\emptyset)$*.*

b. Admissible *(*ADM*)*

$$\text{ADM}(\mathcal{H}) = \{S\ :\ S \textit{ is conflict-free and every } x \in S \textit{ is acceptable to } S\}.$$

c. Preferred *(*PE*)*

$$\text{PE}(\mathcal{H}) = \{S\ :\ S \textit{ is a maximal (w.r.t. } \subseteq\textit{) set in } \text{ADM}(\mathcal{H})\}.$$

d. Stable *(*SE*)*

$$\text{SE}(\mathcal{H}) = \{S\ :\ S \in \text{ADM}(\mathcal{H}) \textit{ and every } x \notin S \textit{ is attacked by } S\}.$$

e. Ideal *(*IDL*) (Dung et al. [16,17])*

$$\text{IDL}(\mathcal{H}) \;=\; \{S \;:\; S \in \text{ADM}(\mathcal{H}) \textit{ and } S \subseteq \cap_{T \in \text{PE}(\mathcal{H})}\, T\}.$$

f. Semi-stable *(*SSE*) (Caminada [3])*

$$\text{SSE}(\mathcal{H}) \;=\; \{S \;:\; S \in \text{ADM}(\mathcal{H}) \textit{ and } \forall\, T\ (S \cup S^{+} \subset T \cup T^{+}) \Rightarrow T \notin \text{ADM}(\mathcal{H})\}.$$

We assume the reader is familiar with the standard complexity classes P, NP, coNP together with classes in the so-called *Polynomial Hierarchy* (PH), in particular Σ_2^p and Π_2^p. We further assume some familiarity with the concept of polynomial-time many-one reducibility between decision problems. An accessible introduction to these may be found in Papadimitriou's text [18].

The class D^p is formed by decision problems L, whose positive instances are characterised as those belonging to $L_1 \cap L_2$ where $L_1 \in$ NP and $L_2 \in$ coNP. The problem SAT-UNSAT whose instances are pairs of 3-CNF formulae $\langle \varphi_1, \varphi_2 \rangle$ accepted if φ_1 is satisfiable *and* φ_2 is unsatisfiable has been shown to be complete for this class [18, p. 413]. We may interpret D^p as those decision problems solvable by a (deterministic) polynomial time algorithm allowed to make at most two calls upon an NP *oracle*. More generally, the complexity class P^{NP} consists of decision problems that can be solved by a (deterministic) polynomial time algorithm provided with access to an NP *oracle* (calls upon which take a single step so that only polynomially many invocations are allowed). An important (presumed) subset of P^{NP} is defined by distinguishing whether oracle calls are *adaptive* – i.e. the exact formulation of the next oracle query may be dependent on the answers received to previous questions – or whether such queries are *non-adaptive*, i.e. the form of the questions to be put to the oracle is predetermined allowing all of these to be performed in parallel. The latter class has been denoted $\text{P}_{||}^{\text{NP}}$ and considered in Wagner [19,20], Jenner and Toran [21]. Under the standard complexity-theoretic assumptions, it is conjectured that,

$$\text{P} \;\subset\; \left\{ \begin{array}{c} \text{NP} \\ \text{coNP} \end{array} \right\} \;\subset\; \text{D}^p \;\subset\; \text{P}_{||}^{\text{NP}} \;\subset\; \text{P}^{\text{NP}} \;\subset\; \left\{ \begin{array}{c} \Sigma_2^p \\ \Pi_2^p \end{array} \right\}$$

3 Decision Questions in AFs

Given an AF, $\mathcal{H}(\mathcal{X}, \mathcal{A})$, and a particular extension based semantics $\mathcal{E}$, e.g. $\mathcal{E}$ could be any of SE (stable), PE (preferred), or SSE (semi-stable), Table 1 describes a number of general decision problems relative to $\mathcal{E}$.

The list above is concerned with properties of AFs with respect to a single extension semantics, $\mathcal{E}$. There are in addition, however, a number of natural problems that relate to the behaviour of frameworks regarding distinct extension semantics. In particular given extension semantics $\mathcal{E}$ and $\mathcal{F}$ the decision problem *Coincident* ($\text{COIN}_{\mathcal{E},\mathcal{F}}$) whose instances $\mathcal{H}(\mathcal{X}, \mathcal{A})$ are accepted if and only if $\mathcal{E}(\mathcal{H}) = \mathcal{F}(\mathcal{H})$. Table 2 and its annotations summarise what is known concerning the computational complexity of the decision problems outlined above w.r.t. to $\mathcal{E} \in \{\text{GND}, \text{ADM}, \text{PE}, \text{SE}, \text{IDL}\}$.

Table 1. Decision Problems in AFs

Problem Name	Instance	Question
Verification ($\text{VER}_{\mathcal{E}}$)	$\mathcal{H}(\mathcal{X},\mathcal{A})$; $S \subseteq \mathcal{X}$	Is $S \in \mathcal{E}(\mathcal{H})$?
Credulous Acceptance ($\text{CA}_{\mathcal{E}}$)	$\mathcal{H}(\mathcal{X},\mathcal{A})$; $x \in \mathcal{X}$	Is there *any* $S \in \mathcal{E}(\mathcal{H})$ for which $x \in S$?
Sceptical Acceptance ($\text{SA}_{\mathcal{E}}$)	$\mathcal{H}(\mathcal{X},\mathcal{A})$; $x \in \mathcal{X}$	Is x a member of *every* $T \in \mathcal{E}(\mathcal{H})$?
Existence ($\text{EXISTS}_{\mathcal{E}}$)	$\mathcal{H}(\mathcal{X},\mathcal{A})$	Is $\mathcal{E}(\mathcal{H})$ *non-empty*?
Non-emptiness ($\text{EXISTS}_{\mathcal{E}}^{\neg\emptyset}$)	$\mathcal{H}(\mathcal{X},\mathcal{A})$	Is there any $S \in \mathcal{E}(\mathcal{H})$ for which $S \neq \emptyset$?
Maximality ($\text{MAX}_{\mathcal{E}}$)	$\mathcal{H}(\mathcal{X},\mathcal{A})$; $S \subseteq \mathcal{X}$	Is S a *maximal* (w.r.t. $\subseteq$) set in $\mathcal{E}(\mathcal{H})$?

Table 2. Computational Complexity w.r.t. $\mathcal{E}$

$\mathcal{E}$	VER	CA	SA	EXIST	$\text{EXISTS}^{\neg\emptyset}$	MAX
GND	P ([5])	P ([5])	P ([5])	Trivial ([5])	P ([5])	P ([5])
ADM	P ([5])	NP-c ([13])	Trivial ([5])	Trivial ([5])	NP-c ([13])	coNP-c ([13])
PE	coNP-c ([13])	NP-c ([13])	Π_2^p-c ([14])	Trivial ([5])	NP-c ([13])	coNP-c ([13])
SE	P ([5])	NP-c ([13])	coNP-c / D^p-c	NP-c ([13])	NP-c ([13])	P
IDL	coNP-c ([15])	$\text{P}_{\|\|}^{\text{NP}}$-c (*) ([15])	Trivial ([17])	Trivial ([17])	$\text{P}_{\|\|}^{\text{NP}}$-c (*) [15])	$\text{P}_{\|\|}^{\text{NP}}$-c (*) ([15])

Remarks

1. For a complexity class C, $C - c$ denotes C-completeness.
2. Cases which are described as "trivial" are either those for which the property in question always holds such as existence of preferred extensions, or for which it never holds, e.g. membership in every ideal set (since the empty set is ideal).
3. The two distinct classifications for SA_{SE} arise from the two possible interpretations of sceptical acceptance w.r.t. stable extensions for AFs without any, i.e. if one regards $x \in \cap_{S \in \text{SE}(\mathcal{H})}\ S$ as holding even when $\text{SE}(\mathcal{H}) = \emptyset$ then the decision problem is coNP–complete ([13] via commentary of [14, p. 189]). If, however, one requires $\mathcal{H}$ to have *at least one* stable extension as a precondition for x to be sceptically accepted the decision problem becomes D^p-complete. While this upper bound is straightforward, space precludes details of the matching lower bound proof.
4. For $\mathcal{E} \in \{\text{GND}, \text{PE}, \text{SE}, \text{SSE}\}$, the verification and maximality problems are equivalent, i.e. $\text{VER}_{\mathcal{E}}(\mathcal{H}, S) \Leftrightarrow \text{MAX}_{\mathcal{E}}(\mathcal{H}, S)$.
5. The cases annotated by (*) for ideal sets make use of *randomized* polynomial time reductions in the style of [22]. Only weaker lower bounds have been obtained by standard deterministic reductions.

4 Computational Complexity of Semi-stable Semantics

In this paper we show that Table 2 may be extended as described in Table 3.

The results described in the first three lines of Table 3 are straightforward developments of constructions originally presented in [13] and [14]. The hardness

Table 3. Computational Complexity w.r.t. Semi-stable extensions

Problem	Lower Bound	Upper Bound	
VER_{SSE}	coNP-hard	coNP	Theorem 1
$\text{EXISTS}^{\neg\emptyset}_{\text{SSE}}$	NP-hard	NP	Corollary 1
$\text{COIN}_{\text{PE,SSE}}$	Π^p_2-hard	Π^p_2	Theorem 2
CA_{SSE}	$\text{P}^{\text{NP}}_{\|\|}$-hard	Σ^p_2	Theorem 3
SA_{SSE}	$\text{P}^{\text{NP}}_{\|\|}$-hard	Π^p_2	Theorem 4

results regarding credulous and sceptical acceptance under semi-stable semantics exploit a technical characterisation of complete problems within $\text{P}^{\text{NP}}_{||}$ due to Chang and Kadin [23]. This introduces the concepts of a language having the properties OP_2 and OP_ω where OP is one of the Boolean operators $\{\text{AND}, \text{OR}\}$.

Definition 3. *([23, pp. 175–76] Let L be a language, i.e. a set of finite words over an alphabet. The languages, $\text{AND}_k(L)$ and $\text{OR}_k(L)$ $(k \geq 1)$ are*

$$\begin{array}{lll} \text{AND}_k(L) & =_{\text{def}} & \{\langle w_1, w_2, \ldots, w_k\rangle \ : \ \forall\ 1 \leq i \leq k\ w_i \in L\} \\ \text{OR}_k(L) & =_{\text{def}} & \{\langle w_1, w_2, \ldots, w_k\rangle \ : \ \exists\ 1 \leq i \leq k\ w_i \in L\} \end{array}$$

The languages $\text{AND}_\omega(L)$ and $\text{OR}_\omega(L)$ are,

$$\text{AND}_\omega(L) \ =_{\text{def}} \ \bigcup_{k\geq 1} \text{AND}_k(L) \quad ; \quad \text{OR}_\omega(L) \ =_{\text{def}} \ \bigcup_{k\geq 1} \text{OR}_k(L)$$

A language, L, is said to have property OP_k (resp. OP_ω) if $\text{OP}_k(L) \ \leq^p_m \ L$ (resp. $\text{OP}_\omega(L) \ \leq^p_m \ L$).

The reason why these language operations are of interest is the following result.

Fact 1. *([23, Thm. 9, p. 182])*
A language L is $\text{P}^{\text{NP}}_{||}$–complete (via $\leq^p_m$ reducibility) if and only if all of the following hold.

F1. $L \in \text{P}^{\text{NP}}_{||}$.
F2. L *is* NP*–hard* and L *is co*NP*–hard.*
F3. L *has property* AND_2.
F4. L *has property* OR_ω.

Theorem 1. VER_{SSE} *is co*NP*–complete.*

Proof. Given $\mathcal{H}(\mathcal{X}, \mathcal{A})$ and $S \subseteq \mathcal{X}$, S defines a semi-stable extension of $\mathcal{H}$ if and only if, S is admissible and

$$\forall\ T \subseteq \mathcal{X} \quad T \in \text{ADM}(\mathcal{H}) \quad \Rightarrow \quad \neg\ (S \cup S^+ \ \subset \ T \cup T^+)$$

a test which is easily accomplished by a coNP algorithm.

For coNP-hardness it suffices to consider the special case $S = \emptyset$, i.e. the problem $\text{VER}_{\text{SSE}}(\mathcal{H}, \emptyset)$, which is the complement of $\text{EXISTS}^{\neg\emptyset}_{\text{SSE}}$. Given an instance of *unsatisfiability* – without loss of generality a 3-CNF formula $\varphi(Z_n) = \wedge_{j=1}^{m} C_j$ – with each C_j a disjunction of literals from $\{z_1, \ldots, z_n, \neg z_1, \ldots, \neg z_n\}$, the AF, $\mathcal{H}_\varphi(\mathcal{X}, \mathcal{A})$ has

$$\begin{aligned} \mathcal{X} &= \{\varphi, C_1, \ldots, C_m\} \cup \{z_i,\ \neg z_i\ :\ 1 \le i \le n\} \\ \mathcal{A} &= \{\langle C_j, \varphi\rangle\ :\ 1 \le j \le m\} \cup \{\langle z_i, \neg z_i\rangle,\ \langle \neg z_i, z_i\rangle\ :\ 1 \le i \le n\} \cup \\ &\quad \{\langle z_i, C_j\rangle\ :\ z_i \text{ occurs in } C_j\} \cup \{\langle \neg z_i, C_j\rangle\ :\ \neg z_i \text{ occurs in } C_j\} \end{aligned}$$

As shown by [13], there is an admissible set containing the argument φ if and and only if $\varphi(Z_n)$ is satisfiable, i.e. $\neg\text{CA}_{\text{ADM}}(\mathcal{H}_\varphi, \varphi)$ if and only if $\varphi(Z_n)$ is unsatisfiable. Modify $\mathcal{H}_\varphi$ to the AF, $\mathcal{K}_\varphi$ as follows: add a single new argument ψ to $\mathcal{X}$ together with $2n + 1$ new attacks $\{\langle \psi, z_i\rangle\ :\ 1 \le i \le n\}$, $\{\langle \psi, \neg z_i\rangle\ :\ 1 \le i \le n\}$, and $\langle \varphi, \psi\rangle$. The AF, $\mathcal{K}_\varphi$ has a non-empty *preferred* extension if and only if the CNF, φ is satisfiable. Hence, $\text{VER}_{\text{SSE}}(\mathcal{K}_\varphi, \emptyset)$ holds if and only if $\varphi(Z_n)$ is unsatisfiable.

Corollary 1. $\text{EXISTS}^{\neg\emptyset}_{\text{SSE}}$ *is* NP*-complete.*

Proof. For membership in NP it suffices to test if $\text{EXISTS}^{\neg\emptyset}_{\text{ADM}}(\mathcal{H})$. The NP-hardness lower bound is immediate from the the proof of Thm. 11.

Theorem 2. $\text{COIN}_{\text{PE,SSE}}$ *is* Π^p_2*–complete.*

Proof. Given $\mathcal{H}(\mathcal{X}, \mathcal{A})$ every preferred extension of $\mathcal{H}$ is also a semi-stable extension if and only if,

$$\forall\ S \subseteq \mathcal{X}\quad S \notin \text{PE}(\mathcal{H})\quad \vee\quad S \in \text{SSE}(\mathcal{H})$$

This may be re-written as, $\forall\ S,\ T\ \exists\ U\ f(S, T, U)$ where $f(S, T, U)$ is the (polynomial time decidable) predicate

$$(S \notin \text{ADM}(\mathcal{H})) \bigvee (U \in \text{ADM}(\mathcal{H})) \wedge (S \subset U)) \bigvee (\ (S \cup S^+ \subset T \cup T^+) \Rightarrow (T \notin \text{ADM}(\mathcal{H})))$$

That is, "for every subset (S), *either* S does not define a preferred extension of $\mathcal{H}$ (by reason of inadmissibility or containment in a larger admissible set, U) *or* (should S be a preferred extension), there is no (admissible) set (T) for which $S \cup S^+$ is strictly contained in $T \cup T^+$".

The test described can be accomplished in Π^p_2.

To establish Π^p_2–hardness we reduce to the complementary problem – i.e. that of deciding if a given $\mathcal{H}$ has a preferred extension which fails to be semi-stable, using the Σ^p_2–complete problem, QSAT^{Σ}_2 instances of which comprise a CNF formula, $\varphi(Y_n, Z_n)$ over disjoint sets of propositional variables, that are accepted if there is some instantiation (α_Y) of Y_n for which every instantiation, (β_Z) of Z_n fails to satisfy $\varphi(Y_n, Z_n)$, i.e. $\exists\ \alpha_Y\ \forall\ \beta_Z\ \neg\varphi(\alpha_Y, \beta_Z)$. Given an

instance, $\varphi(Y_n, Z_n)$, consider the AF $\mathcal{G}_\varphi(\mathcal{W}, \mathcal{B})$ formed from the AF $\mathcal{H}_\varphi(\mathcal{X}, \mathcal{A})$ of Thm. 1, i.e. $\mathcal{X} \subset \mathcal{W}$ and $\mathcal{A} \subset \mathcal{B}$, so that

$$\begin{aligned}
\mathcal{W} &= \{\varphi, C_1, \ldots, C_m\} \cup \{y_i, \neg y_i, z_i, \neg z_i \ : \ 1 \leq i \leq n\} \cup \{b_1, b_2, b_3\} \\
\mathcal{B} &= \{\langle C_j, \varphi\rangle \ : \ 1 \leq j \leq m\} \cup \\
&\quad \{\langle y_i, \neg y_i\rangle, \langle \neg y_i, y_i\rangle, \langle z_i, \neg z_i\rangle, \langle \neg z_i, z_i\rangle \ : \ 1 \leq i \leq n\} \cup \\
&\quad \{\langle y_i, C_j\rangle \ : \ y_i \text{ occurs in } C_j\} \cup \{\langle \neg y_i, C_j\rangle \ : \ \neg y_i \text{ occurs in } C_j\} \cup \\
&\quad \{\langle z_i, C_j\rangle \ : \ z_i \text{ occurs in } C_j\} \cup \{\langle \neg z_i, C_j\rangle \ : \ \neg z_i \text{ occurs in } C_j\} \cup \\
&\quad \{\langle \varphi, b_1\rangle, \langle \varphi, b_2\rangle, \langle \varphi, b_3\rangle, \langle b_1, b_2\rangle, \langle b_2, b_3\rangle, \langle b_3, b_1\rangle\} \cup \\
&\quad \{\langle b_1, z_i\rangle, \langle b_1, \neg z_i\rangle \ : \ 1 \leq i \leq n\}
\end{aligned}$$

From [14] this framework has a *non-stable* preferred extension if and only if $\varphi(Y_n, Z_n)$ is accepted as an instance of QSAT_2^Σ.[1] In particular, every satisfying instantiation of $\varphi(Y_n, Z_n)$ induces a corresponding *stable* extension of $\mathcal{G}_\varphi$. Now, noting that $\varphi(Y_n, Z_n)$ is accepted as instance of QSAT_2^Σ if and only if the CNF, $\psi(Y_n \cup \{u\}, Z_n)$ in which each clause of φ has a new variable u added to it, is also so accepted we claim that the AF, $\mathcal{G}_\psi$ has a preferred (but not semi-stable) extension if and only if there is an instantiation, α of $Y_n \cup \{u\}$ under which $\psi(\alpha, Z_n)$ is unsatisfiable. First suppose $\text{SSE}(\mathcal{G}_\psi) \subset \text{PE}(\mathcal{G}_\psi)$. Since, $u = \top$ satisfies ψ, from the properties of $\mathcal{G}_\psi$ it follows that this has at least one stable extension, hence

$$\text{SE}(\mathcal{G}_\psi) = \text{SSE}(\mathcal{G}_\psi) \subset \text{PE}(\mathcal{G}_\psi)$$

and so $\mathcal{G}_\psi$ must contain a preferred extension which is not stable. From the analysis given in [14] we can construct an instantiation α_Y of Y_n which has $\psi(\alpha_Y, u = \bot, Z_n)$ unsatisfiable.

A similar analysis to that of [14] identifies a (non-stable) preferred extension for any instantation of α of $Y_n \cup \{u\}$ under which $\psi(\alpha, Z_n)$ is unsatisfiable, i.e. if ψ is accepted as an instance of QSAT_2^Σ then the set of preferred extensions of $\mathcal{G}_\psi$ does not coincide with its set of semi-stable extensions.

As a consequence of Fact 1, the lower bounds on CA_{SSE} and SA_{SSE} are derived using the following four part constructions.

S1. Prove that CA_{SSE} (resp. SA_{SSE}) is NP–hard.
S2. Prove that CA_{SSE} (resp. SA_{SSE}) is coNP–hard.
S3. Prove that CA_{SSE} (resp. SA_{SSE}) has property AND_2 (in fact we will show both to have property AND_ω).
S4. Prove that CA_{SSE} (resp. SA_{SSE}) has property OR_ω.

Theorem 3

a. CA_{SSE} *is in* Σ_2^p.
b. CA_{SSE} *is* $\text{P}_{||}^{\text{NP}}$*–hard.*

[1] Each witnessing non-stable but preferred extension is formed by a subset of $\{y_i, \neg y_i \ 1 \leq i \leq n\}$ for which the instantiation, α_Y of the corresponding literals in Y_n to $\top$ results in $\varphi(\alpha_Y, Z_n)$ being unsatisfiable.

Proof. We omit the relatively easy upper bound stated in (a) and concentrate on the second part of the theorem statement. Using the characterisation of $\mathrm{P}_{||}^{\mathrm{NP}}$–complete languages described in Fact 1 the theorem follows given arguments that (S1)–(S4) all hold.

S1. $\mathrm{CA}_{\mathrm{SSE}}$ is NP–hard.
Given an instance, $\varphi(Z_n)$ of SAT form an instance $\langle \mathcal{H}_\varphi, \varphi \rangle$ of $\mathrm{CA}_{\mathrm{SSE}}$ in which $\mathcal{H}_\varphi$ is the AF described in the proof of Thm. 1. The argument φ is in a stable (hence semi-stable) extension of $\mathcal{H}_\varphi$ if and only if $\varphi(Z_n)$ is satisfiable. We deduce that $\mathrm{CA}_{\mathrm{SSE}}$ is NP–hard as a result.

S2. $\mathrm{CA}_{\mathrm{SSE}}$ is coNP–hard.
Given an instance $\varphi(Z_n)$ of UNSAT, first form the AF, $\mathcal{H}_\varphi$ as described in S1. Modify $\mathcal{H}_\varphi$ to a system $\mathcal{K}_\psi$ which has a new argument ψ added together with attacks $\langle \varphi, z_i \rangle$ and $\langle \varphi, \neg z_i \rangle$ for each $1 \leq i \leq n$ and $\{\langle \varphi, \psi \rangle \ \langle \psi, \varphi \rangle\}$. The instance is completed by choosing ψ as the argument of interest. The instance $\langle \mathcal{K}_\psi, \psi \rangle$ is accepted if there is a semi-stable extension containing ψ. If, however, there is a preferred extension containing φ, then this extension is also a stable extension which would preclude membership of ψ in a semi-stable set. Such a preferred extension exists if and only if $\varphi(Z_n)$ is satisfiable, so that $\langle \mathcal{K}_\psi, \psi \rangle$ is accepted as an instance of $\mathrm{CA}_{\mathrm{SSE}}$ if and only if $\varphi(Z_n)$ is *unsatisfiable.*

S3. $\mathrm{CA}_{\mathrm{SSE}}$ has property AND_ω.
Let $\langle \langle \mathcal{H}_1, x_1 \rangle, \langle \mathcal{H}_2, x_2 \rangle, \ldots, \langle \mathcal{H}_k, x_k \rangle \rangle$ define an instance of $\mathrm{AND}_k(\mathrm{CA}_{\mathrm{SSE}})$. Form an instance $\langle \mathcal{H}, z \rangle$ of $\mathrm{CA}_{\mathrm{SSE}}$ in which the k frameworks, $\mathcal{H}_i$ are extended by adding a set of k arguments $\{y_1, \ldots, y_k\}$, an argument z, and attacks $\{\langle y_i, z \rangle, \langle x_i, y_i \rangle\}$ for each $1 \leq i \leq k$. We claim that $\langle \mathcal{H}, z \rangle$ is accepted as an instance of $\mathrm{CA}_{\mathrm{SSE}}$ if and only if each $\langle \mathcal{H}_i, x_i \rangle$ is accepted as such an instance. Suppose the latter is true and that S_i is a semi-stable extension in $\mathcal{H}_i$ that contains x_i. Then $S = \cup_{i=1}^{k} S_i \cup \{z\}$ is certainly admissible (since each attack $\langle y_i, z \rangle$ is countered by the attack $\langle x_i, z \rangle$. Furthermore S is a semi-stable extension as

$$S \cup S^+ = \bigcup_{i=1}^{k} S_i \ \cup \ S_i^+ \ \cup \ \{y_1, y_2, \ldots, y_k, z\}$$

so that all of the new arguments $\{y_1, \ldots, y_k, z\}$ occur within $S \cup S^+$. In total from semi-stable extensions S_i containing x_i we construct a semi-stable extension S containing z.

Conversely suppose S is a semi-stable extension of $\mathcal{H}$ and that $z \in S$. It is certainly the case that $\{x_1, \ldots, x_k\} \subset S$ since this is required in order to defend the attacks $\langle y_i, z \rangle$. Consider the set $S_i = S \cap \mathcal{X}_i$ where $\mathcal{X}_i$ is the set of arguments in $\mathcal{H}_i$. Noting that $x_i \in S_i$ we claim that S_i is a semi-stable extension in $\mathcal{H}_i$. Suppose this were not the case so that some admissible subset, T of $\mathcal{X}_i$ satisfies $S_i \cup S_i^+ \subset T_i \cup T_i^+$. Without loss of generality we may assume $x_i \notin T$ so that $x_i \in T^+$. Now consider the set $R = S \backslash (S_i \cup \{z\}) \cup T_i \cup \{y_i\}$. Observe that R is admissible: y_i being defended

by the argument attacking x_i in T_i. In addition, however,

$$\begin{array}{rcl} S \cup S^+ & = & \cup_{j=1}^{k}\ S_j\ \cup\ S_j^+\ \ \cup\ \ \{y_1, \ldots, y_k, z\} \\ & \subset & \cup_{j\neq i}^{k}\ S_j \cup S_j^+\ \ T_i \cup T_i^+\ \ \cup\ \ \{y_1, \ldots, y_k, z\} \\ & = & R \cup R^+ \end{array}$$

with R admissible and $z \notin R$: this contradicts the premise that S is semi-stable.

S4. CA_{SSE} has property OR_ω.
Let $\langle\langle \mathcal{H}_1, x_1\rangle, \langle \mathcal{H}_2, x_2\rangle, \ldots, \langle \mathcal{H}_k, x_k\rangle\rangle$ define an instance of $\text{OR}_k(\text{CA}_{\text{SSE}})$. Form an instance $\langle \mathcal{H}, z\rangle$ of CA_{SSE} in which the k frameworks, $\mathcal{H}_i$ are extended by adding arguments $\{y, z\}$ and attacks $\{\langle x_i, y\rangle\ :\ 1 \leq i \leq k\}$ and $\langle y, z\rangle$. First suppose, without loss of generality the $x_1 \in S_1$ a semi-stable extension of $\mathcal{H}_1$. Let $\langle S_2, \ldots, S_k\rangle$ be semi-stable extensions of $\mathcal{H}_i$ for $2 \leq i \leq k$. Then it easily follows that $S = \{z\} \cup \cup_{i=1}^{k} S_i$ is not only admissible but a semi-stable of $\mathcal{H}$ containing z. On the other hand suppose S with $x \in S$ is a semi-stable extension of $\mathcal{H}$. There must be at least one $\mathcal{H}_i$ for which $x_i \in S$ in order to defend the attack by y on z. Now considering the set $S \cap S_i$ gives a semi-stable extension in $\mathcal{H}_i$ containing x_i by a similar argument to that used in S3.

Theorem 4

a. SA_{SSE} *is in* Π_2^p.
b. SA_{SSE} *is* $\text{P}_{||}^{\text{NP}}$*–hard.*

Proof. (Outline) We again omit the easy upper bound proof, concentrating on (b). As before we obtain the result in 4 stages.

T1. SA_{SSE} is NP–hard.
Given an instance $\varphi(Z_n)$ of satisfiability, form the framework $\mathcal{K}_\varphi$ described in Thm 1. The instance of SA_{SSE} is given by $\langle \mathcal{K}_\varphi, \varphi\rangle$. If $\varphi(Z_n)$ is satisfiable then the subset $\langle a_1, a_2, \ldots, a_n\rangle$ of $\{z_i,\ \neg z_i\ :\ 1 \leq i \leq n\}$ indicated by any satisfying assignment together with φ is a stable extension and hence also semi-stable. Hence $\varphi(Z_n)$ satisfiable implies $\text{SA}_{\text{SSE}}(\mathcal{K}_\varphi, \varphi)$ holds. On the other hand if $\varphi(Z_n)$ is unsatisfiable then (as argued in the proof of Thm. 1) $\mathcal{K}_\varphi$ has only the empty set as a semi-stable extension. We deduce that SA_{SSE} is NP–hard.

T2. SA_{SSE} is coNP–hard.
Given an instance, $\varphi(Z_n)$ of unsatisfiability, construct the framework $\mathcal{K}_\varphi$ described above but without the attacks $\langle \psi, z_i\rangle$ and $\langle \psi, \neg z_i\rangle$. The instance of SA_{SSE} is $\langle \mathcal{K}_\varphi, \psi\rangle$. If $\varphi(Z_n)$ is satisfiable then ψ cannot belong to the semi-stable extension induced by a satisfying instantiation (since this contains the argument φ). On the other hand, if $\varphi(Z_n)$ is unsatisfiable then every stable extension of $\mathcal{K}_\varphi$ has the form: exactly one of each of the pairs $\{z_i, \neg z_i\}$, the subset of clause arguments which are unattacked; and the argument ψ. Hence ψ is a member of every semi-stable extension if and only if $\varphi(Z_n)$ is unsatisfiable.

T3. SA_{SSE} has property AND_ω. Similar to (S3).

T4. SA_{SSE} has property OR_ω. Similar to (S4).

5 Conclusions and Further Work

The main aim of this paper has been to initiate a detailed study of computational complexity within semi-stable semantics the eventual goal being to have an understanding of complexity issues in this model comparable to that demonstrated in earlier work on complexity in Dung's model, e.g. [13,14,24,15]. Although we have been able to demonstrate non-trivial lower bounds for the six fundamental decision problems described in Table 3, a number of questions remain unresolved. By far the most significant of these concern the gaps between $\mathrm{P}_{||}^{\mathrm{NP}}$-hardness for deciding credulous or sceptical acceptance of an argument under semi-stable semantics and the Σ_2^p (resp. Π_2^p) upper bounds for these problems.

There are some (admittedly far from conclusive) indications that the credulous acceptance problem is within $\mathrm{P}_{||}^{\mathrm{NP}}$, i.e. that Σ_2^p is an overestimate. For example, we observe that of the semantics specified in Defn. 2 (a)–(e), the problem $\mathrm{CA}_{\mathcal{E}}$ is decidable in P ($\mathcal{E} = \mathrm{GND}$) or NP ($\mathcal{E} \in \{\mathrm{ADM},\mathrm{PE},\mathrm{SE}\}$); even for the notionally "hardest" case – $\mathcal{E} = \mathrm{IDL}$ – the upper bound is $\mathrm{P}_{||}^{\mathrm{NP}}$. Thus, in the event of our $\mathrm{P}_{||}^{\mathrm{NP}}$-hardness failing to be optimal, deciding credulous acceptance within this semantics becomes "significantly harder" than the analogous problems in any of the semantics proposed to date. The labelling techniques of Verheij [25] and Caminada [12] may well offer useful starting points from which to construct $\mathrm{P}_{||}^{\mathrm{NP}}$ methods for $\mathrm{CA}_{\mathrm{SSE}}$. The challenge is to develop an argument labelling scheme $\Lambda : \mathcal{X} \rightarrow \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ with the following properties: given $\langle \mathcal{H}(\mathcal{X}, \mathcal{A}), x, \lambda \rangle$ the question "is there a Λ labelling of $\mathcal{H}$ in which $\Lambda(x) = \lambda$?" is decidable in NP; and, given a correct Λ-labelling of $\mathcal{H}$, membership of x in a semi-stable extension can be decided in polynomial time. We note that the $\mathrm{P}_{||}^{\mathrm{NP}}$ algortihm to decide $\mathrm{CA}_{\mathrm{IDL}}$ from [15] can be interpreted as using exactly this approach with $\Lambda : \mathcal{X} \rightarrow \{I, O\}$ and $\Lambda(x) = I$ if and only if $\mathrm{CA}_{\mathrm{ADM}}(\mathcal{H}, x)$. In contrast, by analogy with the known complexity of $\mathrm{SA}_{\mathrm{PE}}$ proven in [14], we conjecture that the Π_2^p upper bound for the sceptical variant is exact.

As stated in the introduction, one of the aims of this paper was to examine at what computational cost the advantages of semi-stable over stable semantics are obtained. Despite the absence of exact bounds for the credulous and sceptical acceptance variants it is clear from the proven lower bounds that some non-trivial increase in complexity is incurred: these problems being NP-complete (resp. coNP-complete/D^p-complete) for stable semantics but $\mathrm{P}_{||}^{\mathrm{NP}}$-hard in the semi-stable case; the verification problem is in P (stable) but coNP-complete (semi-stable). Deciding if there is a (non-empty) semi-stable extension (resp. *any* stable extension) are, however, of equivalent complexity both being NP-complete. In order fully to assess this cost would, of course, require further progress to be made on exact classification. We note, however, that if $\mathrm{CA}_{\mathrm{SSE}} \in \mathrm{P}_{||}^{\mathrm{NP}}$ (as conjectured above) there is a case for regarding the computational cost of testing credulous acceptance in semi-stable semantics as "similar" to that of $\mathrm{CA}_{\mathrm{SE}}$: in informal terms the class $\mathrm{P}_{||}^{\mathrm{NP}}$ is arguably "closer" to NP than to Σ_2^p. In particular, if a suitable labelling approach can be adopted, the computational cost of implementing this is that of solving $|\mathcal{X}|$ independent NP-complete problems as opposed to a single such problem.

Finally it would be of interest to consider complexity-theoretic aspects of semi-stable semantics within the assumption-based frameworks (ABFs) of Bondarenko *et al* [26]. While an important core of such results considering the generic decision problems of Table 1 with respect to preferred and stable semantics, is presented in work of Dimopoulos *et al* [27,28,29] similar treatments for semi-stable (and indeed ideal) semantics have yet to be developed.

References

1. Verheij, B.: Two approaches to dialectical argumentation: admissible sets and argumentation stages. In: Meyer, J.J., van der Gaag, L. (eds.) Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC 1996), pp. 357–368. Utrecht University, Utrecht (1996)
2. Verheij, B.: Deflog: on the logical interpretation of prima facie justified assumptions. Journal of Logic and Computation 13, 319–346 (2003)
3. Caminada, M.: Semi-stable semantics. In: Dunne, P.E., Bench-Capon, T.J.M. (eds.) Proc. 1st Int. Conf. on Computational Models of Argument. FAIA, vol. 144, pp. 121–130. IOS Press, Amsterdam (2006)
4. Caminada, M., Ben-Naim, J.: Postulates for paraconsistent reasoning and fault tolerant logic programming. Technical Report Technical Report UU-CS-2007-004, Utrecht University (2007)
5. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reason, logic programming, and N-person games. Artificial Intelligence 77, 321–357 (1995)
6. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the 5th International Conference/Symposium on Logic Programming, pp. 1070–1080. MIT Press, Cambridge (1988)
7. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9(3/4), 365–385 (1991)
8. Arieli, O., Avron, A.: The value of four values. Artificial Intellligence 102, 97–141 (1998)
9. Reiter, R.: A logic for default reasoning. Artificial Intelligence 13, 81–132 (1980)
10. Baroni, P., Giacomin, M.: On principle-based evaluation of extension-based argumentation semantics. Artificial Intelligence 171, 675–700 (2007); special issue on argumentation in artificial intelligence
11. Baroni, P., Giacomin, M.: Comparing argumentation semantics with respect to skepticism. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 210–221. Springer, Heidelberg (2007)
12. Caminada, M.: An algorithm for computing semi-stable semantics. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 222–234. Springer, Heidelberg (2007)
13. Dimopoulos, Y., Torres, A.: Graph theoretical structures in logic programs and default theories. Theoretical Computer Science 170, 209–244 (1996)
14. Dunne, P.E., Bench-Capon, T.J.M.: Coherence in finite argument systems. Artificial Intelligence 141, 187–203 (2002)
15. Dunne, P.E.: The computational complexity of ideal semantics I: abstract argumentation frameworks. In: Proc. 2nd Int. Conf. on Computational Models of Argument. FAIA, vol. 172, pp. 147–158. IOS Press, Amsterdam (2008)

16. Dung, P.M., Mancarella, P., Toni, F.: A dialectical procedure for sceptical assumption-based argumentation. In: Dunne, P.E., Bench-Capon, T.J.M. (eds.) Proc. 1st Int. Conf. on Computational Models of Argument. FAIA, vol. 144, pp. 145–156. IOS Press, Amsterdam (2006)
17. Dung, P.M., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artificial Intelligence 171, 642–674 (2007)
18. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading (1994)
19. Wagner, K.: Bounded query computations. In: Proc. 3rd Conf. on Structure in Complexity Theory, pp. 260–277 (1988)
20. Wagner, K.: Bounded query classes. SIAM Jnl. Comput. 19, 833–846 (1990)
21. Jenner, B., Toran, J.: Computing functions with parallel queries to NP. Theoretical Computer Science 141, 175–193 (1995)
22. Valiant, L.G., Vazirani, V.V.: NP is as easy as detecting unique solutions. Theoretical Computer Science 47, 85–93 (1986)
23. Chang, R., Kadin, J.: On computing Boolean connectives of characteristic functions. Math. Syst. Theory 28, 173–198 (1995)
24. Dunne, P.E.: Computational properties of argument systems satisfying graph-theoretic constraints. Artificial Intelligence 171, 701–729 (2007)
25. Verheij, B.: A labelling approach to the computation of credulous acceptance in argumentation. In: Proc. IJCAI 2007, pp. 623–628 (2007)
26. Bondarenko, A., Dung, P., Kowalski, R., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artificial Intelligence 93, 63–101 (1997)
27. Dimopoulos, Y., Nebel, B., Toni, F.: Preferred arguments are harder to compute than stable extensions. In: Thomas, D. (ed.) Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999, pp. 36–43. Morgan Kaufmann, San Francisco (1999)
28. Dimopoulos, Y., Nebel, B., Toni, F.: Finding admissible and preferred arguments can be very hard. In: Cohn, A.G., Giunchiglia, F., Selman, B. (eds.) KR2000: Principles of Knowledge Representation and Reasoning, pp. 53–61. Morgan Kaufmann, San Francisco (2000)
29. Dimopoulos, Y., Nebel, B., Toni, F.: On the computational complexity of assumption-based argumentation for default reasoning. Artificial Intelligence 141, 55–78 (2002)

Query Answering in the Description Logic Horn-$\mathcal{SHIQ}$*

Thomas Eiter[1], Georg Gottlob[2], Magdalena Ortiz[1], and Mantas Šimkus[1]

[1] Institute of Information Systems, TU Vienna, Austria
{eiter,ortiz,simkus}@kr.tuwien.ac.at
[2] Computing Laboratory, Oxford University, United Kingdom
georg.gottlob@comlab.ox.ac.uk

Abstract. We provide an ExpTime algorithm for answering conjunctive queries (CQs) in Horn-$\mathcal{SHIQ}$, a Horn fragment of the well-known Description Logic $\mathcal{SHIQ}$ underlying the OWL-Lite standard. The algorithm employs a domino system for model representation, which is constructed via a worst-case optimal tableau algorithm for Horn-$\mathcal{SHIQ}$; the queries are answered by reasoning over the domino system. Our algorithm not only shows that CQ answering in Horn-$\mathcal{SHIQ}$ is not harder than satisfiability testing, but also that it is polynomial in data complexity, making Horn-$\mathcal{SHIQ}$ an attractive expressive Description Logic.

1 Introduction

Driven by the development of semantically enhanced systems, as in the context of the Semantic Web and of Enterprise Application Integration, query answering in Description Logics (DLs) has emerged as an important topic. A variety of algorithms have been proposed for this problem and, aiming at different applications, aspects like *combined* and *data complexity* have been guiding their development. The former characterizes the cost of query answering in the general case, while the latter in the case when the query and the knowledge base except the factual part are fixed. Data complexity is especially important for applications in which DLs are used to formalize rich data models for data repositories, as in such context the model is static as compared to the data contents, and typical user queries are known. For querying DLs, conjunctive queries (CQs) have been most widely considered and three major settings have been addressed:

- Very expressive DLs for which standard reasoning tasks, like satisfiability testing or instance checking, are intractable both in data and combined complexity. As query answering is at least as hard, the problem is trivially intractable in general. For example, CQs in $\mathcal{SHIQ}$ have 2ExpTime-complete combined complexity [13] and coNP-complete data complexity [6].

* This work was partially supported by the Austrian Science Fund (FWF) grant P20840, Wolfgang Pauli Institute (WPI), and the Mexican National Council for Science and Technology (CONACYT) grant 187697.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 166–179, 2008.

- Tailored DLs, like DL-Lite [5], which aim at lower complexity at the price of limited expressiveness. In DL-Lite, CQ answering is coNP-complete in combined complexity, but polynomial if the query is fixed, and has very low data complexity (reducible to FOL, thus inside logarithmic space).
- Weak DLs, like $\mathcal{EL}$ [1], for which standard reasoning and CQ answering under data complexity are P-complete, while CQ answering under combined complexity is intractable [17]. Several extensions of $\mathcal{EL}$ sharing this property (e.g., $\mathcal{ELH}$, $\mathcal{ELI}^f$, $\mathcal{EL}^+$, $\mathcal{EL}^{++}$) can be found in [17,12,10].

Since CQ answering is intractable under combined complexity already over very simple knowledge bases, fragments of expressive DLs with tractable data complexity are of particular interest. Ideally, such fragments should also allow for CQ answering with combined complexity not higher than of standard reasoning.

In this paper, we identify Horn-$\mathcal{SHIQ}$ as a DL with this property. It was introduced in [9] as a Horn fragment of $\mathcal{SHIQ}$, in which the syntax is restricted in a way that disjunction is not expressible. While standard reasoning in Horn-$\mathcal{SHIQ}$ is ExpTime-complete in general [11], it is polynomial if the taxonomy is fixed [9].

Our main contributions and results are briefly summarized as follows:

- We provide an ExpTime algorithm for answering CQs in Horn-$\mathcal{SHIQ}$. The algorithm is based on answering tree-shaped queries over particular trees (that capture a universal model of the KB) finitely represented by domino systems. The latter are relatives of saturated mosaic sets known in other branches of logic, and the recent knot sets [16] and domino sets [18] in DLs.
- For constructing domino systems, we exploit a dedicated tableaux-based algorithm for consistency checking in Horn-$\mathcal{SHIQ}$, which is of independent interest. It adapts the standard $\mathcal{SHIQ}$ tableaux [8] (using, e.g., *anywhere blocking* [15] and a kind of *lazy unfolding* [7]) to terminate in deterministic single exponential time, yielding a representation of a universal model of $\mathcal{K}$ such that each CQ over $\mathcal{K}$ can be answered on it. This may also be exploited to precompile $\mathcal{K}$ into a (query-independent) domino system for on-line query answering.
- Based on our algorithm, we show that CQs in Horn-$\mathcal{SHIQ}$ have ExpTime-complete combined complexity and P-complete data complexity. We also present a fragment of Horn-$\mathcal{SHIQ}$ for which CQs are easier. In Horn-$\mathcal{SHQ}^-$, which forbids inverse roles and existential projection on the left hand side of containment axioms, the combined complexity of CQs is lowered to PSpace-completeness.

As Horn-$\mathcal{SHIQ}$ is an expressive fragment of OWL-Lite, our results are relevant for the Semantic Web context. They extend in fact from CQs to the class of positive (existential) queries. Our result on the combined complexity of CQs in Horn-$\mathcal{SHIQ}$ is of particular interest. Firstly, it reveals another expressive DL for which CQ answering is not harder than standard reasoning (cf. [16,13,14]). Secondly, it suggests that the exponential jump in combined complexity of CQs by adding inverse roles to $\mathcal{ALC}$, which was found by Lutz [13], relies on their interaction with disjunction. In other words, if disjunction is eliminated, then inverse roles do not make CQ answering harder.

2 Preliminaries

The Description Logics $\mathcal{SHIQ}$ and Horn-$\mathcal{SHIQ}$. We assume countably infinite sets $\mathbf{C}$, $\mathbf{R}$ and $\mathbf{I}$ of *concept names*, *role names*, and *individuals* respectively, where $\mathbf{C}$ contains special concepts names $\top$ and $\bot$. *Roles* are expressions R and R^-, where $R \in \mathbf{R}$, and their inverses are $Inv(R) = R^-$ and $Inv(R^-) = R$, respectively. For any roles R and S, $R \sqsubseteq S$ is a *role inclusion axiom (RI)*, and $Trans(R)$ is a *transitivity axiom (TA)*. For any set $\mathcal{R}$ of RIs and TAs, $\sqsubseteq^*_\mathcal{R}$ denotes the reflexive transitive closure of $\{(R, S) \mid R \sqsubseteq S \in \mathcal{R} \text{ or } Inv(R) \sqsubseteq Inv(S) \in \mathcal{R}\}$; we write $Trans_\mathcal{R}(R')$ if $R' \sqsubseteq^*_\mathcal{R} R$ and $R \sqsubseteq^*_\mathcal{R} R'$ for some R s.t. $Trans(R) \in \mathcal{R}$ or $Trans(R^-) \in \mathcal{R}$. A role R is *simple* w.r.t. $\mathcal{R}$, if there is no $S \sqsubseteq^*_\mathcal{R} R$ with $Trans_\mathcal{R}(S)$.

Concepts are inductively defined as follows: (a) each $A \in \mathbf{C}$ is a concept, and (b) if C, D are concepts, R is a role, and S is a simple role, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$, $\geq n\, S.C$ and $\leq n\, S.C$, for $n \geq 1$, are concepts.

An expression $C \sqsubseteq D$, where C, D are concepts, is a *general concept inclusion axiom (GCI)*, and expressions $a{:}A$ and $\langle a, b\rangle{:}R$, where $A \in \mathbf{C}$, a and b are individuals, and R is a role, are *concept* and *role assertions*, respectively. A $\mathcal{SHIQ}$ *knowledge base* (KB) is a tuple $\mathcal{K}{=}\langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$, where the *TBox* $\mathcal{T}$ is a finite set of GCIs, the *RBox* $\mathcal{R}$ is a finite set of RIs and TAs, and the *ABox* $\mathcal{A}$ is a finite nonempty set of assertions. We denote by $\mathbf{C}(\mathcal{K})$, $\mathbf{R}(\mathcal{K})$ and $\mathbf{I}(\mathcal{K})$ the sets of concept names, role names, and individuals occurring in $\mathcal{K}$.

An *interpretation* $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I}\rangle$ for a KB $\mathcal{K}$ consists of a nonempty *domain* $\Delta^\mathcal{I}$ and a *valuation function* $\cdot^\mathcal{I}$ that maps each individual $c \in \mathbf{I}(\mathcal{K})$ to an element $c^\mathcal{I} \in \Delta^\mathcal{I}$, each concept name $C \in \mathbf{C}(\mathcal{K})$ to a subset $C^\mathcal{I}$ of $\Delta^\mathcal{I}$, and each role name $R \in \mathbf{R}(\mathcal{K})$ to a subset $R^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$, in such a way that $\top^\mathcal{I} = \Delta^\mathcal{I}$ and $\bot^\mathcal{I} = \emptyset$. The function $\cdot^\mathcal{I}$ is extended to all concepts and roles in the standard way (see, e.g., [11]), and satisfaction of $\mathcal{K}$ by $\mathcal{I}$ ($\mathcal{I} \models \mathcal{K}$), i.e. modelhood, is also standard.

The DL Horn-$\mathcal{SHIQ}$ was introduced [9] as a fragment of $\mathcal{SHIQ}$. The main idea is to restrict the syntax in a way that $\sqcup$ is not expressible, establishing a correspondence to a Horn fragment of first-order logic with equality. Without loss of generality, we focus here on a normal form of Horn-$\mathcal{SHIQ}$ in [11], to which each Horn-$\mathcal{SHIQ}$ KB can be efficiently rewritten while preserving the answers to arbitrary CQs (as follows from [11]).

Definition 1. *(Normal)* Horn-$\mathcal{SHIQ}$ *KBs contain only GCIs of the forms*

$$A \sqcap B \sqsubseteq C \qquad A \sqsubseteq \forall R.B \qquad A \sqsubseteq \geq m\, S.B$$
$$\exists R.A \sqsubseteq B \qquad A \sqsubseteq \exists R.B \qquad A \sqsubseteq \leq 1\, S.B$$

where A, B, C are concept names, R is a role, S is a simple role, and $m \geq 1$.

Example 1. Assume two Horn-$\mathcal{SHIQ}$ KBs $\mathcal{K}_1 = \langle \mathcal{T}, \emptyset, \mathcal{A}\rangle$ and $\mathcal{K}_2 = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$, where $\mathcal{T} = \{A \sqsubseteq \exists R.A, B \sqsubseteq \exists P.C\}$, $\mathcal{R} = \{P \sqsubseteq R\}$, and $\mathcal{A} = \{a : A, a : B\}$. Note that both are consistent, however adding $A \sqsubseteq \forall R.\bot$ to $\mathcal{T}$ makes them inconsistent.

Conjunctive Queries. Let $\mathbf{V}$ be a countably infinite set of variables. A *(Boolean) conjunctive query* (CQ, or *query*) over a KB $\mathcal{K}$ is a finite set q of atoms of the

form $A(x)$ or $R(x,y)$, where A is a concept name, R is a role and $x, y \in \mathbf{V}$.[1] By $\mathbf{V}(q)$ we denote the variables occurring in the atoms of q. The query graph of q is the directed graph G^q over nodes $\mathbf{V}(q)$ with an edge between nodes x and y iff $R(x,y) \in q$ for some role R. The query q is *tree-shaped* if G^q is a tree.

A *match for q in an interpretation $\mathcal{I}$* is a mapping $\theta : \mathbf{V}(q) \to \Delta^{\mathcal{I}}$ s.t. (i) $\theta(x) \in A^{\mathcal{I}}$ for each $A(x) \in q$, and (ii) $\langle \theta(x), \theta(y) \rangle \in R^{\mathcal{I}}$ for each $R(x,y) \in q$. We say that $\mathcal{I}$ satisfies q ($\mathcal{I} \models q$), if q has a match in $\mathcal{I}$, and that $\mathcal{K}$ entails q ($\mathcal{K} \models q$), if q has a match in each model $\mathcal{I}$ of $\mathcal{K}$.

Example 2. Assume the queries $tq_1 = \{A(x), R(x,y), A(y), R(x,z), C(z)\}$ and $tq_2 = \{B(x), R(x,y), A(y), P(x,z), C(z)\}$. As easily seen, $\mathcal{K}_1 \not\models tq_1$ and $\mathcal{K}_1 \models tq_2$, while $\mathcal{K}_2 \models tq_1$ and $\mathcal{K}_2 \models tq_2$. Note that both queries are tree-shaped.

3 Tree Queries over Domino Trees

This section describes an algorithm for answering CQs over trees induced by *domino systems*, which is exploited in the next sections for deciding CQ entailment in Horn-$\mathcal{SHIQ}$ KBs. A domino system finitely represents a possibly infinite tree-shaped interpretation that be can built by connecting matching *dominoes*.

Definition 2. *A* domino *is a tuple $\langle c, r, c' \rangle$ where c, c' are sets of concepts names and r is a set of roles (w.r.t. an underlying alphabet). A* domino system *is a tuple $\langle D, \rhd, \mathcal{R} \rangle$, where D is a set of dominoes, $\rhd \subseteq D \times D$ is a* direct successor *relation with $c'_1 = c_2$ whenever $\langle c_1, r_1, c'_1 \rangle \rhd \langle c_2, r_2, c'_2 \rangle$, and $\mathcal{R}$ is an RBox. We also require that for each $\langle c, r, c' \rangle \in D$, the set r is closed under role inclusions in $\mathcal{R}$, i.e., $R \in r$ and $R \sqsubseteq R' \in \mathcal{R}$ imply $R' \in r$. Furthermore, D contains one designated* initial domino *of the form $\langle \emptyset, \emptyset, c' \rangle$.*

Following the terminology in [14], we define next the tree-shaped interpretation induced by a domino system. Its domain is represented by a prefix-closed set of words; for a word $w = e_1 \cdots e_n$, let $\langle w | e_{n+1} \rangle$ denote the word $e_1 \cdots e_n \cdot e_{n+1}$.

Definition 3. *The* tree base *of a domino system $\mathcal{D} = \langle D, \rhd, \mathcal{R} \rangle$ is the interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ (w.r.t. the alphabet underlying $\mathcal{D}$) defined as follows:*

1. *The domain $\Delta^{\mathcal{I}}$ is the smallest set of words over dominoes such that:*
   - *if $t \in D$ is the initial domino, then $t \in \Delta^{\mathcal{I}}$;*
   - *if $t_1 \cdots t_n \in \Delta^{\mathcal{I}}$ and $t_n \rhd t_{n+1}$, then $t_1 \cdots t_n \cdot t_{n+1} \in \Delta^{\mathcal{I}}$.*
2. *The valuation function $\cdot^{\mathcal{I}}$ is defined as follows:*
   - *For each atomic concept A, $A^{\mathcal{I}} = \{\langle s | t \rangle \in \Delta^{\mathcal{I}} \mid t = \langle c, r, c' \rangle \wedge A \in c'\}$.*
   - *For each role name R,*

$$R^{\mathcal{I}} = \{(s, \langle s|t \rangle) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid t = \langle c, r, c' \rangle \wedge R \in r\} \cup \{(\langle s|t \rangle, s) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid t = \langle c, r, c' \rangle \wedge Inv(R) \in r\}.$$

[1] W.l.o.g, no individuals occur in q; we can replace each a in q by a new variable y, add $C_a(y)$ to q and $a : C_a$ to $\mathcal{A}$, where C_a is a new concept name. Non-Boolean queries (i.e., with answer variables) can be reduced to Boolean queries as usual.

function: checkRoleSucc
input: $\mathcal{D} = \langle D, \rhd, \mathcal{R}\rangle$; dominoes $t_1 = \langle c_1, r_1, c_1'\rangle$, $t_3 = \langle c_3, r_3, c_3'\rangle$ from D; role set $r \neq \emptyset$;
output: *true* iff t_3 is an r-successor of t_1
if t_1 has no direct successor in $\mathcal{D}$, **then return** $false$;
Initialize t with some direct successor $\langle c_2, r_2, c_2'\rangle$ of t_1;
$s := r_2$; $i := 0$
repeat
  if $t = t_3$ and $r \subseteq s$ **then return** *true*;
  if $r \not\subseteq s$ or t has no direct successor in $\mathcal{D}$, **then return** $false$;
  Reassign t to some direct successor $\langle c_2, r_2, c_2'\rangle$ of t;
  Set s' to the smallest set closed under the following rule:
  if $S \sqsubseteq^*_{\mathcal{R}} R$, $S \in s$, $S \in r_2$, *and* $Trans_{\mathcal{R}}(S)$, *then* $R \in s'$;
  $s := s'$
until $i > |D|$; **return** $false$

Fig. 1. Verifying Successor Dominoes

The domino tree $\mathcal{T}_{\mathcal{D}} = \langle \Delta^{\mathcal{T}}, \cdot^{\mathcal{T}}\rangle$ *of* $\mathcal{D}$ *is the interpretation identical to* $\mathcal{I}$ *except that, for each role* R, *we have* $R^{\mathcal{T}} = R^{\mathcal{I}} \cup \bigcup_{S \sqsubseteq^*_{\mathcal{R}} R \wedge Trans_{\mathcal{R}}(S)} (S^{\mathcal{I}})^+$.

Query entailment in a domino system is naturally defined via the existence of matches in the represented domino tree. We first provide a procedure to verify the existence of special *ordered* matches for tree-shaped queries, and we then extend the result to all CQs via the standard method of query treeification.

Definition 4. *A domino system* $\mathcal{D}$ entails *a CQ* q *(*$\mathcal{D} \models q$*), if there is a match for* q *in* $\mathcal{T}_{\mathcal{D}}$. *A match* π *for a tree-shaped CQ* tq *in* $\mathcal{T}_{\mathcal{D}}$ *is* ordered *if, for each* $x, y \in \mathbf{V}(tq)$, $\pi(x)$ *is a proper prefix of* $\pi(y)$ *whenever* $R(x, y) \in tq$ *for some* R. *We write* $\mathcal{D} \models_o tq$, *if there is some ordered match for* tq *in* $\mathcal{T}_{\mathcal{D}}$.

Let tq be a fixed tree-shaped query and $\mathcal{D} = \langle D, \rhd, \mathcal{R}\rangle$ a domino system with tree $\mathcal{T}_{\mathcal{D}}$. To obtain a procedure for deciding $\mathcal{D} \models_o tq$, we provide some necessary and sufficient conditions for the existence of ordered matches that can be verified without building $\mathcal{T}_{\mathcal{D}}$ explicitly. Roughly, we search for an association of dominoes from $\mathcal{D}$ with the variables of tq. As the association must witness an ordered match, the domino t_x associated with variable x must encode the concept names needed to satisfy each unary atom $A(x) \in tq$, while for each role atom $R(x, y) \in tq$, the domino t_x must 'reach' the domino t_y via an R-path.

Definition 5. *For two dominoes* $t_1 = \langle c_1, r_1, c_1'\rangle$ *and* $t_3 = \langle c_3, r_3, c_3'\rangle$ *in* D *and a set of roles* $r \neq \emptyset$, *we say* t_3 *is a* r-successor *of* t_1 *if one of the following holds:*

(a) $t_1 \rhd t_3$ *and* $r \subseteq r_3$, *or*
(b) for some role set r', D *contains an* r'*-successor* t_2 *of* t_1 *such that* $t_2 \rhd t_3$ *and for each* $R \in r$ *there exists* $S \in r'$ *with* $Trans_{\mathcal{R}}(S)$, $S \sqsubseteq^*_{\mathcal{R}} R$ *and* $S \in r_3$.

We are ready to define domino associations, which characterize the $\models_o$ relation.

Definition 6. *A* domino association for tq *is a mapping* μ *that assigns to each* $z \in \mathbf{V}(tq)$ *a domino* $\mu(z) \in D$ *in a way such that, for each pair* x, $y \in \mathbf{V}(tq)$*:*

(a) if $A(x) \in tq$, then $A \in c'$, where $\mu(x) = \langle c, r, c' \rangle$; and
(b) if $r = \{R \mid R(x,y) \in tq\}$ is not empty, then $\mu(y)$ is an r-successor of $\mu(x)$.

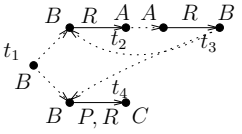


Fig. 2. Domino $\mathcal{D}$

Example 3. Consider the domino system $\mathcal{D} = \langle D, \triangleright, \mathcal{R} \rangle$ in Figure 2, where the black arrows show the dominoes of D and the dashed arrows the $\triangleright$ relation; $\mathcal{R} = \{Trans(R)\}$ and the initial domino is t_1. A possible domino association for tq_1 in Example 2 is the mapping μ_1 with $\mu_1(x) = t_2$, $\mu_1(y) = t_2$, $\mu_1(z) = t_4$. Note that t_2 and t_4 are $\{R\}$-successors of t_2 as R is transitive, and that tq_1 has no domino association for $\mathcal{R} = \emptyset$. The query tq_2 in Example 2 has a domino association even in this case, witnessed by μ_2 with $\mu_2(x) = t_3$, $\mu_2(y) = t_2$, and $\mu_2(z) = t_4$.

The following is immediate from the definition of $\models_o$ and Definition 6.

Theorem 1. *$\mathcal{D} \models_o tq$ iff there exists a domino association for tq.*

By Theorem 1, we can decide $\mathcal{D} \models_o tq$ by deciding existence of a domino association. We exploit for the latter the procedure checkRoleSucc in Figure 1, which nondeterministically checks whether a domino t_2 is an r-successor of a domino t_1.

Proposition 1. *Let t_1, t_2 be dominoes of $\mathcal{D}$, and r a role set. Then t_2 is an r-successor of t_1 iff some run of* checkRoleSucc$(\mathcal{D}, t_1, t_2, r)$ *returns true.*

Now the following simple procedure assocDominoes$(\mathcal{D}, tq)$ non-deterministically decides the existence of a domino association for tq w.r.t. $\mathcal{D}$: (1) guess a mapping μ from $\mathbf{V}(tq)$ to dominoes of $\mathcal{D}$, and (2) check satisfaction of the conditions (a) and (b) in Definition 6; to check (b), call checkRoleSucc for each arc in tq.

Theorem 2. *$\mathcal{D} \models_o tq$ iff some run of* assocDominoes$(\mathcal{D}, tq)$ *returns true.*

Having a procedure to decide $\mathcal{D} \models_o tq$ for tree-shaped queries tq, we now settle deciding $\mathcal{D} \models q$ for arbitrary CQs q. Following [4,6,14], we consider query treeifications, i.e., tree-shaped queries whose matches induce matches for q.[2]

Definition 7. *For every CQ q, let $q^{\mathcal{R}}$ be the smallest query such that: (a) $q \subseteq q^{\mathcal{R}}$, (b) $R(x,y) \in q^{\mathcal{R}}$ and $R \sqsubseteq P \in \mathcal{R}$ implies $P(x,y) \in q^{\mathcal{R}}$, (c) $R(x,y) \in q^{\mathcal{R}}$, $R(y,z) \in q^{\mathcal{R}}$ and $Trans_{\mathcal{R}}(R)$ imply $R(x,z) \in q^{\mathcal{R}}$, and (d) $R(x,y) \in q^{\mathcal{R}}$ implies $Inv(R)(y,x) \in q^{\mathcal{R}}$. A* treeification *of q is a tree-shaped query q' such that $|q'| \leq 2|q|$ and there exists a mapping θ from $\mathbf{V}(q)$ to $\mathbf{V}(q')$ fulfilling*

a) $A(x) \in q$ implies $A(\theta(x)) \in q'$, and
b) $R(x,y) \in q$ implies $R(\theta(x), \theta(y)) \in (q')^{\mathcal{R}}$.

[2] Unlike [4,6,14], we do not use treeifications to reduce CQ entailment to concept satisfiability, as this would require the use of role conjunction and the decidability of this extension of Horn-$\mathcal{SHIQ}$ in ExpTime is not apparent.

As easily shown, each match for a treeification q' of q in $\mathcal{T}_\mathcal{D}$ is also a match for q. On the other hand, from each match for q in $\mathcal{T}_\mathcal{D}$, we can obtain a treeification q' that is mappable into $\mathcal{T}_\mathcal{D}$ via an ordered match.[3] Hence, we obtain:

Theorem 3. *For any CQ q, $\mathcal{D} \models q$ iff $\mathcal{D} \models_o q'$, for some treeification q' of q.*

It follows that we can decide $\mathcal{D} \models q$ by listing all treeifications of q and posing them separately over $\mathcal{D}$. Note that there are finitely many treeifications of q.

4 A Tableaux Algorithm for Horn-$\mathcal{SHIQ}$

In order to exploit the results of Section 3 for query answering in Horn-$\mathcal{SHIQ}$, we first provide a tableau algorithm for KB satisfiability. Like the standard $\mathcal{SHIQ}$ tableau [8], it uses a *completion forest* to represent a model; in the next section we extract from it a domino system that can be used for query answering.

In what follows, $\mathcal{K}=\langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$ is a Horn-$\mathcal{SHIQ}$ KB. We use A, B, C to denote concepts names; D, E to denote (arbitrary) concepts; R, R' to denote a role; S a simple role; and a, b to denote individuals.

Most of the following definitions are based on [8], while 9 and 14 follow [15] and are related to *anywhere blocking*. Definition 8 is simplified since only normalized KBs are considered, and the $\approx$ relation from [8] is omitted in Definition 10.[4]

Definition 8. *(concept closure) We define $\mathbf{Cl}(\mathcal{K})$ as the smallest set of concepts closed under subconcepts such that (i) $D, E \in \mathbf{Cl}(\mathcal{K})$ for every $D \sqsubseteq E \in \mathcal{T}$; and (ii) if $\forall R.A \in \mathbf{Cl}(\mathcal{K})$, $Trans_\mathcal{R}(R')$ and $R' \sqsubseteq^*_\mathcal{R} R$ for some R', then $\forall R'.A \in \mathbf{Cl}(\mathcal{K})$.*

Definition 9. *((named/unnamed) nodes) We assume a countably infinite set $\mathbf{N}$ of* nodes *and a strict total order $\prec$ on $\mathbf{N}$. Each $a \in \mathbf{I}(\mathcal{K})$ is associated with one fixed node $n_a \in \mathbf{N}$; the nodes n_a are* named*, all other nodes are* unnamed*.*

Definition 10. *(completion forest) A* completion forest for a KB $\mathcal{K}$ *is a tuple $\mathcal{F} = \langle \mathcal{N}, \mathcal{E}, \mathcal{L}, \not\approx\rangle$ where $\mathcal{N} \subseteq \mathbf{N}$ and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ define a directed graph; $\mathcal{L}$ is a labeling function assigning each $n \in \mathcal{N}$ a subset of $\mathbf{Cl}(\mathcal{K})$ and each pair $u, u' \in \mathcal{N} \times \mathcal{N}$ to a set of roles (over $\mathcal{K}$), in such a way that $\mathcal{L}(u, u') = \emptyset$ for all $(u, u') \notin \mathcal{E}$; and $\not\approx \subseteq \mathcal{N} \times \mathcal{N}$ is a binary relation, tacitly assumed to be symmetric.*

Definition 11. *(successor, neighbor) For a completion forest $\mathcal{F}=\langle \mathcal{N}, \mathcal{E}, \mathcal{L}, \not\approx\rangle$ and a pair $u, u' \in \mathcal{N}$, u' is a* successor *of u if $(u, u') \in \mathcal{E}$. The inverse of successor is called* predecessor*; the transitive closures of successor and predecessor are* ancestor *and* descendant *respectively. For all R, u' is an R-*successor *of u if $R' \in \mathcal{L}(u, u')$ for some R' with $R' \sqsubseteq^*_\mathcal{R} R$. We call u' an R-*neighbor *of u, if u' is an R-*successor *of u, or if u is an $Inv(R)$-*successor *of u'.*

[3] As implicit in [6], such a q' with $|q'| \leq 2|q|$ exists: to obtain a treeification from a match, one replaces each atom $R(x, y) \in q$ with a pair of atoms in case x, y are mapped (i) to the same node, or (ii) to nodes in different branches of the domino tree.

[4] It is irrelevant for query answering, but could be emulated e.g. using node labels C_a.

Definition 12. *(clash-free completion forest) A completion forest $\mathcal{F}$ contains a* clash*, if* (i) *for some $u \in \mathcal{N}$, $\bot \in \mathcal{L}(u)$; or* (ii) *for some $u \in \mathcal{N}$ with $\leq n\, S.C \in \mathcal{L}(u)$, u has $n+1$ S-neighbors $w_0, \ldots, w_n$ such that, for all $0 \leq i < j \leq n$, $C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j \in \mathcal{F}$. If $\mathcal{F}$ contains no clash, then $\mathcal{F}$ is* clash-free*.*

Definition 13. *(initial completion forest) The* initial completion forest $\mathcal{F}_\mathcal{A}$ *for $\mathcal{K}$ has the named node n_a labeled with $\mathcal{L}(n_a) = \{A \in \mathbf{C}(\mathcal{K}) \mid a : A \in \mathcal{A}\}$ for each individual $a \in \mathbf{I}(\mathcal{K})$, and an edge $(n_a, n_b) \in \mathcal{E}$ labeled $\mathcal{L}(n_a, n_b) = \{R \mid \langle a, b\rangle : R \in \mathcal{A}\}$ for each pair $a, b \in \mathbf{I}(\mathcal{K})$; the relation $\not\approx$ is empty.*

Definition 14. *(blocking) For a completion forest $\mathcal{F} = \langle \mathcal{N}, \mathcal{E}, \mathcal{L}, \not\approx \rangle$, a node $u \in \mathcal{N}$ is* blocked *if u is unnamed and u is either directly or indirectly blocked; u is* indirectly blocked *if one of its ancestors is blocked; u is* directly blocked *if none of its ancestors is blocked and there is some $u' \lessdot u$ such that u, u' are unnamed nodes, $\mathcal{L}(u) = \mathcal{L}(u')$, $\mathcal{L}(v) = \mathcal{L}(v')$, and $\mathcal{L}(v, u) = \mathcal{L}(v', u')$, where v and v' are the predecessors of u and u' respectively.*

The expansion rules are given in Figure 3, where a node $u \in \mathbf{N}$ is *new* in $\mathcal{F}$ if $u' \lessdot u$ for every $u' \in \mathcal{N}$. The $\leq$-rule calls the operation *merge*(u, N) described in Figure 4. The rules are similar to those in [8], except for the first three, which (lazily) ensure the satisfaction of the TBox axioms. Also, the restricted form of at-most number restrictions allows us to have just one $\leq$-rule and a deterministic *merge*(u, N) that simultaneously merges all nodes in N into one.[5]

The initial $\mathcal{F}_\mathcal{A}$ is expanded by exhaustively applying the rules in Figure 3. The expansion stops, if a clash is reached; otherwise, it continues until the forest is *complete*, i.e., no rule is applicable. It can be shown similarly as in [8,15] that this algorithm is a decision procedure for KB satisfiability in Horn-$\mathcal{SHIQ}$.

Theorem 4. *Let $\mathcal{K}$ be a* Horn-$\mathcal{SHIQ}$ *KB. Then $\mathcal{K}$ is satisfiable iff a complete and clash-free completion forest for $\mathcal{K}$ can be obtained.*

Note that after applying any rule from Figure 3, the resulting forest is uniquely determined up to renaming of nodes. The only source of differences in the resulting forests lies in possibly different orderings of rule applications (this could be eliminated, e.g., using $\lessdot$ and any fixed ordering on $\mathbf{Cl}(K)$ and on the rules). However, these differences are not relevant: each $\mathcal{F}$ represents a *universal* model $\mathcal{I}_\mathcal{F}$ (defined as its standard unravelling [8]) that is embeddable into every model of $\mathcal{K}$, and can be used for query answering. The following is shown by a straightforward induction on the construction of $\mathcal{I}_\mathcal{F}$:

Theorem 5. *Let $\mathcal{I}$ be a model of $\mathcal{K}$, let $\mathcal{F}$ be a complete and clash-free completion forest for $\mathcal{K}$, and let $\mathcal{I}_\mathcal{F}$ be the model of $\mathcal{K}$ represented by $\mathcal{F}$. Then there is a homomorphic embedding of $\mathcal{I}_\mathcal{F}$ into $\mathcal{I}$. Hence, for any CQ q, $\mathcal{K} \models q$ iff $\mathcal{I}_\mathcal{F} \models q$.*

[5] Note that the TBox internalization of [8] is not adequate for Horn-$\mathcal{SHIQ}$, and that the other rules of [8] are not necessary due to the normal form of the KB.

$\mathcal{T}$-rule:	if	$A \sqsubseteq D \in \mathcal{T}$, $A \in \mathcal{L}(u)$, and $D \notin \mathcal{L}(u)$,
	then	$\mathcal{L}(u) := \mathcal{L}(u) \cup \{D\}$.
$\mathcal{T}_\sqcap$-rule:	if	$A \sqcap B \sqsubseteq C \in \mathcal{T}$, $\{A, B\} \subseteq \mathcal{L}(u)$, u is not indirectly blocked, and $C \notin \mathcal{L}(u)$,
	then	$\mathcal{L}(u) := \mathcal{L}(u) \cup \{C\}$.
$\mathcal{T}_\exists$-rule:	if	$\exists R.A \sqsubseteq B \in \mathcal{T}$, $B \notin \mathcal{L}(u)$, u is not indirectly blocked, and u has an R-neighbor u' with $A \in \mathcal{L}(u')$,
	then	$\mathcal{L}(u) := \mathcal{L}(u) \cup \{B\}$.
$\exists$-rule:	if	$\exists R.A \in \mathcal{L}(u)$, u is not blocked, and u has no R-neighbor u' with $A \in \mathcal{L}(u')$,
	then	set $\mathcal{N} = \mathcal{N} \cup \{u'\}$, $\mathcal{E} = \mathcal{E} \cup \{(u, u')\}$, $\mathcal{L}(u, u') := \{R\}$ and $\mathcal{L}(u') := \{A\}$ for some u' new in $\mathcal{F}$.
$\forall$-rule:	if	$\forall R.A \in \mathcal{L}(u)$, u is not indirectly blocked, and u has an R-neighbour u' with $A \notin \mathcal{L}(u')$,
	then	$\mathcal{L}(u') := \mathcal{L}(u') \cup \{A\}$.
$\forall_+$-rule:	if	$\forall R.A \in \mathcal{L}(u)$, u is not indirectly blocked, there is some R' with $\mathit{Trans}_\mathcal{R}(R')$ and $R' \sqsubseteq^*_\mathcal{R} R$, and there is an R'-neighbour u' of u with $\forall R'.A \notin \mathcal{L}(u')$,
	then	$\mathcal{L}(u') := \mathcal{L}(u') \cup \{\forall R'.A\}$.
$\geq$-rule:	if	$\geq m\, S.A \in \mathcal{L}(u)$, u is not blocked, and there are no m S-neighbours $u_1, \ldots, u_m$ of u such that $A \in \mathcal{L}(u_i)$ and $u_i \not\approx u_j$ for $1 \leq i < j \leq m$,
	then	set $\mathcal{N} = \mathcal{N} \cup \{u_1, \ldots, u_m\}$, $\mathcal{E} = \mathcal{E} \cup \{(u, u_1), \ldots, (u, u_m)\}$, $\mathcal{L}(u, u_i) := \{S\}$, $\mathcal{L}(u_i) := \{A\}$ and $u_i \not\approx u_j$ for $1 \leq i < j \leq m$ and $u_1, \ldots, u_m$ new in $\mathcal{F}$.
$\leq$-rule:	if	$\leq 1\, S.A \in \mathcal{L}(u)$, u is not indirectly blocked, N is the set of all S-neighbours u' of u with $A \in \mathcal{L}(u')$, $\lvert N\rvert > 1$ and there is no pair u', u'' in N with $u' \not\approx u''$,
	then	$\mathit{merge}(u, N)$.

Fig. 3. Tableaux expansion rules

5 Conjunctive Queries over Horn-$\mathcal{SHIQ}$

To answer CQs over Horn-$\mathcal{SHIQ}$ KBs, we exploit the method for answering tree-shaped queries over domino systems. For this section, we assume that $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ is a consistent Horn-$\mathcal{SHIQ}$ KB, and q is an arbitrary CQ.[6] From a complete and clash-free completion forest $\mathcal{F}_\mathcal{K}$ for $\mathcal{K}$, we extract a domino system $\mathcal{D}_{\mathcal{F}_\mathcal{K}}$ that encodes a forest-shaped universal model of $\mathcal{K}$ for query answering. We then rewrite q into a set of tree-shaped queries which can be posed separately over $\mathcal{D}_{\mathcal{F}_\mathcal{K}}$, such that $\mathcal{K} \models q$ iff one of the generated queries is entailed by $\mathcal{D}_{\mathcal{F}_\mathcal{K}}$.

The transformation of the completion forest into $\mathcal{D}_{\mathcal{F}_\mathcal{K}}$, which we now present, eliminates the 'graph part' of the forest by encoding it into the initial domino.

Definition 15. *Let $\mathcal{F} = \langle \mathcal{N}, \mathcal{E}, \mathcal{L}, \not\approx \rangle$ be a complete and clash-free completion forest for $\mathcal{K}$. For every $u \in \mathcal{N}$, let $\mathcal{L}'(u) = \mathcal{L}(u) \cap \mathbf{C}(\mathcal{K})$.*

[6] Note that in case of inconsistent KBs, query entailment is trivial.

(1) let u_0 be the $\preccurlyeq$-minimal element of N;
(2) let $N' = N \setminus \{u_0\}$; let N'' be the minimal set containing N',
each unnamed successor u' of a node in N', and all descendants of u';
(3) if $(u', n) \in \mathcal{E}$ for some $u' \in N'$ and some named n,
then $\mathcal{E} := \mathcal{E} \cup (u_0, n)$ and $\mathcal{L}(u_0, n) := \mathcal{L}(u_0, n) \cup \mathcal{L}(u', n)$;
if $(n, u') \in \mathcal{E}$ for some $u' \in N'$ and some named n,
then $\mathcal{E} := \mathcal{E} \cup (n, u_0)$ and $\mathcal{L}(n, u_0) := \mathcal{L}(n, u_0) \cup \mathcal{L}(n, u')$;
(4) set $\mathcal{N} := \mathcal{N} \setminus N''$, $\mathcal{E} := \mathcal{E} \setminus \{(v, u') \mid u' \in N''\}$, restrict $\mathcal{L}$ and $\not\approx$ to the new $\mathcal{N}$, $\mathcal{E}$;
(5) add $u_0 \not\approx v$ for every $v \in \mathcal{N}$ such that $u' \not\approx v$ for some $u' \in N'$;
(6) set $\mathcal{L}(u_0) := \mathcal{L}(u_0) \cup \mathcal{L}(N')$, where $\mathcal{L}(N') = \bigcup_{u' \in N'} \mathcal{L}(u')$;
(7) if $(u_0, u) \in \mathcal{E}$ then $\mathcal{L}(u_0, u) := \mathcal{L}(u_0, u) \cup \mathcal{L}(N', u)$, else $\mathcal{L}(u, u_0) := \mathcal{L}(u, u_0) \cup \mathcal{L}(u, N')$,
where $\mathcal{L}(u, N') = \bigcup_{u' \in N} \mathcal{L}(u, u')$ and $\mathcal{L}(N', u) = \{Inv(R) \mid R \in \mathcal{L}(u, N')\}$.

Fig. 4. The *merge*(u, N) operation on $\mathcal{F} = \langle \mathcal{N}, \mathcal{E}, \mathcal{L}, \not\approx \rangle$

Let $t_0 = \langle \emptyset, \emptyset, c \rangle$ be the domino where c is the smallest set of fresh concept names such that Root $\in c$ and, for each pair n_a, n_b of named nodes in $\mathcal{N}$, (i) $A \in \mathcal{L}'(n_a)$ implies $A_a \in c$, (ii) if n_b is an R-neighbour of n_a, then $R_{a,b} \in c$, (iii) $R_{a,b} \in c$ and $R \sqsubseteq R' \in \mathcal{R}$ implies $R'_{a,b} \in c$, (iv) $R_{a,b} \in c$, $R_{b,d} \in c$ and $Trans_{\mathcal{R}}(R)$ implies $R_{a,d} \in c$, and (v) $R_{a,b} \in c$ implies $Inv(R)_{b,a} \in c$.

For each named node $n_a \in \mathcal{N}$, let t_a denote the domino $t_a = \langle c, \{Q_a\}, c' \rangle$, where Q_a is fresh and $c' = \mathcal{L}'(n_a)$. For a pair $(u, u') \in \mathcal{E}$, let $t(u, u')$ denote the domino $\langle \mathcal{L}'(u), \{R \mid u' \text{ is an } R\text{-neighbour of } u\}, \mathcal{L}'(u') \rangle$. Then $\mathcal{D}_{\mathcal{F}} = \langle D, \rhd, \mathcal{R} \rangle$ is the domino system with initial domino t_0, where

- D is the smallest domino set containing (i) t_0, (ii) t_a for each named $n_a \in \mathcal{N}$, and (iii) each $t(u, u')$ such that $(u, u') \in \mathcal{E}$ and u' is unnamed and not blocked.
- $\rhd$ is the smallest relation s.t. (i) for all named $n_a \in \mathcal{N}$, $t_0 \rhd t_a$ and $t_a \rhd t(n_a, u)$ for every $t(n_a, u) \in D$; and (ii) if $t(u, u'), t(u', v) \in D$ for some $(u, u'), (u', u'') \in \mathcal{E}$ such that either $u'' = v$ is not blocked or u'' is blocked by v, then $t(u, u') \rhd t(u', v)$.

Since the specific complete and clash-free $\mathcal{F}$ does not matter, we assume in what follows a fixed arbitrary $\mathcal{F}_{\mathcal{K}}$ and denote its domino system $\mathcal{D}_{\mathcal{F}_{\mathcal{K}}}$ simply by $\mathcal{D}_{\mathcal{K}}$. As easily seen, we can reconstruct a universal model of $\mathcal{K}$ from the domino tree of $\mathcal{D}_{\mathcal{K}}$. However, for querying $\mathcal{D}_{\mathcal{K}}$, we need to rewrite q in order to handle the links between individuals encoded as concept names in the initial domino.

Definition 16. *A* link rewriting *of q w.r.t. $\mathcal{K}$ is a CQ obtained from q as follows:*

1. *Exhaustively replace, one by one, $R(y, x)$ by $Inv(R)(x, y)$ whenever there are atoms of the form $R(y, x)$ and $S(x, y)$ in q.*
2. *Let $\mu : \mathbf{V}(q) \to \mathbf{I}(\mathcal{K})$ be a partial function, and let $\nu(x) \in \{r \text{ (root)}, i \text{ (inside)}\}$ be a choice for each $x \in dom(\mu)$. Let $\{z\} \cup \{x' \mid x \in \mathbf{V}(q)\}$ be fresh variables. Then, for each $R(x, y) \in q$ with $\{x, y\} \subseteq dom(\mu)$, let $S \sqsubseteq^*_{\mathcal{R}} R$ be arbitrary such that $Trans_{\mathcal{R}}(S)$ holds if either $\nu(x) = i$ or $\nu(y) = i$, and (i) replace $R(x, y)$ in q by $Root(z)$, $S_{a,b}(z)$, where $\mu(x) = a$ and $\mu(y) = b$, and (ii) add in q, depending on the choice $[\nu(x), \nu(y)]$, the following atoms:*

$[r,r]$: $Q_a(z,x)$, $Q_b(z,y)$; $[i,i]$: $Q_a(z,x')$, $Q_b(z,y')$, $S(x,x')$, $S(y',y)$;
$[i,r]$: $Q_a(z,x')$, $Q_b(z,y)$, $S(x,x')$; $[r,i]$: $Q_a(z,x)$, $Q_b(z,y')$, $S(y',y)$.

Roughly speaking, possible R-connections between matches for x and y in $\mathcal{I}_{\mathcal{F}_\mathcal{K}}$ via two individuals a, b are reflected in the query by the atoms $Root(z), S_{a,b}(z)$ and $[\nu(x), \nu(y)]$. For example, if $\mathcal{R} = \{Trans(S), S \sqsubseteq R\}$, the CQ $q' = \{Root(z), S_{a,b}(z), Q_a(z,x'), S(x,x'), A(x), Q_b(z,y), B(y)\}$ is a link rewriting of $q = \{A(x), R(x,y), B(y)\}$, obtained by choosing $\mu(x) = a$, $\mu(y) = b$, $\nu(x) = i$, $\nu(y) = r$ and S. A match for q' in $\mathcal{T}_{\mathcal{D}_\mathcal{K}}$ corresponds to a match for q in $\mathcal{I}_{\mathcal{F}_\mathcal{K}}$ mapping x to a descendant of a (i.e., *inside* the tree rooted at a) and y to b (i.e., the *root* of b's tree), which are connected via an S-path and thus in the extension of R. Note that, as $\nu(x) = i$ was chosen, the non-transitive R can not link the matches of x and y. Choosing $\mu(x) = a$, $\mu(y) = b$, $\nu(x) = r$, $\nu(y) = r$ and R we obtain a rewriting $q'' = \{Root(z), R_{a,b}(z), Q_a(z,x), A(x), Q_b(z,y), B(y)\}$ that captures the matches for q which map x and y to a and b if they are R-neighbors in $\mathcal{I}_{\mathcal{F}_\mathcal{K}}$.

Theorem 6. *$\mathcal{K} \models q$ iff $\mathcal{D}_\mathcal{K} \models q'$ for some link rewriting q' of q. Hence, due to Theorem 3, $\mathcal{K} \models q$ iff $\mathcal{D}_\mathcal{K} \models_o tq$, for some treeification tq of a link rewriting of q.*

Theorem 6 suggests a procedure for deciding $\mathcal{K} \models q$: it suffices to verify the existence of a treeification of a link rewriting of q that has an ordered match in the domino tree of $\mathcal{D}_\mathcal{K}$. The latter can be verified using the method from Section 3.

6 Computational Complexity

We now show that CQ entailment in Horn-$\mathcal{SHIQ}$ is decidable in exponential time. The presented method relies on the extraction of a domino system from a complete and clash-free completion forest. Hence, the following is important.

Theorem 7. *The tableau algorithm for Horn-$\mathcal{SHIQ}$ in Section 4 decides consistency of Horn-$\mathcal{SHIQ}$ KBs in single exponential time. For a consistent KB, it constructs a complete and clash-free completion forest of at most exponential size.*

Proof (Sketch). Definition 14 ensures that if a completion forest $\mathcal{F}$ contains two pairs of nodes with the same node-arc-node label combination, one of them is blocked. The number of such combinations, and thus of nodes in a forest, is single exponential in the input KB $\mathcal{K}$ (in fact, it is bounded by $2^{2|\mathbf{Cl}(\mathcal{K})| \times 2|\mathbf{R}(\mathcal{K})|}$). Using the usual arguments [8], it can be shown that the number of rule applications needed to generate $\mathcal{F}$ is polynomially bounded by the maximal number of nodes it can have, as the shrinking rules do not cause repeated rule applications. □

We are ready to formulate the main complexity results of this paper.

Theorem 8. *Conjunctive query entailment $\mathcal{K} \models q$ in Horn-$\mathcal{SHIQ}$ is* ExpTime*-complete in combined complexity, i.e., in the size of the KB $\mathcal{K}$ and the query q.*

Proof (Sketch). By Theorem 7, we can check the consistency of $\mathcal{K}$ using the tableau-based algorithm in exponential time. If $\mathcal{K}$ is found inconsistent, then $\mathcal{K} \models q$ trivially holds. Otherwise, we can extract the domino system $\mathcal{D}_\mathcal{K}$ from the completion forest $\mathcal{F}_\mathcal{K}$ that was constructed, in time polynomial in $|\mathcal{F}_\mathcal{K}|$.

Each link rewriting q' of q, as well as each treeification tq of q', has size polynomial w.r.t. $|\mathcal{K}| + |q|$. The are at most exponentially many q' and, for each q', at most exponentially many tq; hence, there are at most exponentially many tq in total, and they can be traversed in polynomial space. To show ExpTime membership of $\mathcal{K} \models q$, it is thus sufficient to show that $\mathcal{D}_\mathcal{K} \models_o tq$ is decidable in exponential time w.r.t. $|\mathcal{K}| + |q|$. Indeed, checkRoleSucc runs in NPSpace w.r.t. $|\mathcal{K}| + |q|$ if $\mathcal{D}_\mathcal{K}$ is precomputed (note that the counter i needs only polynomial space). The procedure assocDominoes runs in NP (w.r.t. $|\mathcal{K}| + |q|$) using checkRoleSucc as an oracle. Hence, $\mathcal{D}_\mathcal{K} \models_o tq$ is in $\text{NP}^{\text{NPSpace}} = \text{PSpace}$ w.r.t. $|\mathcal{K}| + |q|$, if $\mathcal{D}_\mathcal{K}$ is precomputed. As computing $\mathcal{D}_\mathcal{K}$ is feasible in exponential time, it follows that deciding $\mathcal{K} \models q$ is in ExpTime. The matching lower bound follows from the ExpTime-hardness of consistency checking in Horn-$\mathcal{SHIQ}$ [11]. □

The next result shows that CQs in Horn-$\mathcal{SHIQ}$ are tractable in data complexity.

Theorem 9. *Conjunctive query entailment $\mathcal{K} \models q$ in* Horn-$\mathcal{SHIQ}$ *is* P*-complete in data complexity, i.e., in the size of the ABox $\mathcal{A}$ of the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$.*

Proof (Sketch). As in Theorem 8, we can check the consistency of $\mathcal{K}$ when we construct the completion forest $\mathcal{F}_\mathcal{K}$. As $\mathcal{T}$ and $\mathcal{R}$ are fixed, $|\mathcal{F}_\mathcal{K}|$ is polynomial w.r.t. $\mathcal{A}$, so $\mathcal{F}_\mathcal{K}$ and $\mathcal{D}_\mathcal{K}$ can be constructed in time polynomial w.r.t. $\mathcal{A}$. Next, for fixed q, $\mathcal{T}$ and $\mathcal{R}$, there are polynomially many treeifications tq of link rewritings q' of q w.r.t. $\mathcal{A}$, and they can be traversed in polynomial time. By Theorem 1, it remains to show that the existence of a domino association $\mu : \mathbf{V}(tq) \to D$, where D is the domino set of $\mathcal{D}_\mathcal{K}$, is decidable in polynomial time w.r.t. $\mathcal{A}$. Since $|\mathbf{V}(tq)|$ is bounded by a constant w.r.t. $\mathcal{A}$ and $|D|$ is polynomial w.r.t. $\mathcal{A}$, there are polynomially many candidate μ w.r.t. $\mathcal{A}$. We can check r-successorship between dominoes t_1, t_2 of $\mathcal{D}_\mathcal{K}$ in time polynomial in $|\mathcal{D}_\mathcal{K}|$, i.e., polynomial w.r.t. $\mathcal{A}$. Hence, we can check whether μ satisfies Definition 6 in polynomial time w.r.t. $\mathcal{A}$. The resulting P membership bound is tight, as consistency checking in any DL allowing for conjunction on the left hand side and quantified universal restrictions on the right hand side of GCIs is P-hard in data complexity [3]. □

The source of ExpTime-hardness of consistency testing in Horn-$\mathcal{SHIQ}$, and hence of query entailment, are inverse roles and concepts of the form $\exists R.A$ on the left hand side of the GCIs. Intuitively, both constructs allow to propagate information from a node to its ancestors in a completion forest; any one of them allows for an encoding of a generic Alternating PSpace Turing machine. If we disallow both, obtaining the DL Horn-$\mathcal{SHQ}^-$, consistency testing and CQ entailment drop to PSpace-completeness. Roughly, the direct successors of a node in a completion forest can be inferred in polynomial time from its label. Hence, the existence of a complete and clash-free completion forest $\mathcal{F}$ is refutable in PSpace without building it, by non-deterministically following a path in $\mathcal{F}$ (of

at most exponential length) that leads to a clash. CQ entailment is decidable in PSPACE by supplying checkRoleSucc with a PSPACE oracle for navigating the domino system $\mathcal{D}_\mathcal{K}$ (note that the explicit construction of $\mathcal{D}_\mathcal{K}$ may require exponential space). This procedure is worst-case optimal, as consistency checking in Horn-$\mathcal{SHQ}^-$ is PSPACE-hard (provable, e.g., by a generic Turing machine encoding). Finally, also our P-completeness result for data complexity of CQs carries over from Horn-$\mathcal{SHIQ}$ to Horn-$\mathcal{SHQ}^-$. Details will be given in the full paper.

7 Related Work and Conclusion

As found recently, answering CQs in some expressive DLs, including $\mathcal{ALCH}$ [16] and $\mathcal{ALCHQ}$ [14], is not harder than standard reasoning, and in fact EXPTIME-complete in combined complexity. Horn-$\mathcal{SHIQ}$ is another such DL but orthogonal to those mentioned, as it offers transitive and inverse roles but excludes disjunction (we note that one can infer from [2] the EXPTIME-completeness result for the DL Horn-$\mathcal{ALCHI}$, i.e., Horn-$\mathcal{SHIQ}$ without transitive roles and number restrictions). Moreover, the data complexity of CQs is polynomial in Horn-$\mathcal{SHIQ}$ but intractable in $\mathcal{ALCH}$ and $\mathcal{ALCHQ}$ (in fact, it is CONP-hard already for $\mathcal{AL}$ [3]).

Different approaches have been recently used to show that CQs have tractable data complexity in some DLs. A large class of such DLs are extensions of $\mathcal{EL}$ [1], considered e.g. in [17,12,10], of which $\mathcal{ELH}$, $\mathcal{ELI}^f$, and $\mathcal{EL}^{++}$ are particularly noticeable. For $\mathcal{EL}$ and $\mathcal{ELH}$, which are subsumed by Horn-$\mathcal{SHIQ}$ but not by Horn-$\mathcal{SHQ}^-$ (due to the absence of existential restrictions on the LHS of the GCIs), a reduction to Datalog has been given in [17]. In both $\mathcal{EL}$ and $\mathcal{ELH}$, CQs have CONP-complete combined complexity and P-complete data complexity. For $\mathcal{ELI}^f$, which is also strictly subsumed by Horn-$\mathcal{SHIQ}$ (as the latter offers qualified universal quantification on the RHS of axioms and more general number restrictions) an explicit (partial) construction of a universal model was used in [10]. Like in Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SHQ}^-$, CQs have P-complete data complexity in $\mathcal{ELI}^f$. Finally, for $\mathcal{EL}^{++}$, which is orthogonal to both Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SHQ}^-$ ($\mathcal{EL}^{++}$ has nominals and regular role hierarchies, but lacks universal quantification), special proof-graphs with automata were used in [12]. Noticeably, CQs in $\mathcal{EL}^{++}$ have PSPACE-complete combined and P-complete data complexity respectively, and thus the same complexity as in Horn-$\mathcal{SHQ}^-$.

Another prominent family for which data complexity has been deeply investigated is DL-Lite [5]. For the core DL-Lite and its extension with functionality and conjunction, which is subsumed by Horn-$\mathcal{SHIQ}$ but not by Horn-$\mathcal{SHQ}^-$, query rewriting into first-order logic over relational databases has been been employed. CQ answering has very low data complexity (inside logarithmic space), and its CONP-complete combined complexity is also much lower than for Horn-$\mathcal{SHQ}^-$.

Our ongoing and future work is devoted to the following issues. The first concerns richer query syntax. As the normal form and the universal model property of Horn-$\mathcal{SHIQ}$ carry over to unions of CQs and the more general positive existential queries (PQs), our results can be immediately extended to them. In fact, answering PQs in Horn-$\mathcal{SHIQ}$ is reducible to answering at most exponentially

many (in the size of the PQ) CQs. Further, since the universal model property allows us to precompile a knowledge base $\mathcal{K}$ into a (query-independent) domino system $\mathcal{D}_\mathcal{K}$ for on-line query answering, the identification of cases in which $\mathcal{D}_\mathcal{K}$ is small would be beneficial. Finally, an obvious issue is a detailed study of other fragments of Horn-$\mathcal{SHIQ}$ besides Horn-$\mathcal{SHQ}^-$. The effect of syntactic restrictions similarly as in [3] on data complexity is here of particular interest.

References

1. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: IJCAI 2003, pp. 325–330 (2003)
2. Cali, A., Gottlob, G., Kifer, M.: Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. In: KR 2008 (to appear, 2008)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: KR 2006 (2006)
4. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: PODS 1998, pp. 149–158 (1998)
5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-lite* family. J. Autom. Reasoning 39(3), 385–429 (2007)
6. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive Query Answering for the Description Logic $\mathcal{SHIQ}$. In: IJCAI 2007, pp. 399–404 (2007)
7. Horrocks, I.: Optimising Tableaux Decision Procedures for Description Logics. PhD thesis, University of Manchester (1997)
8. Horrocks, I., Sattler, U.: A tableaux decision procedure for $\mathcal{SHOIQ}$. In: IJCAI 2005, pp. 448–453 (2005)
9. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In: IJCAI 2005, pp. 466–471 (2005)
10. Krisnadhi, A., Lutz, C.: Data complexity in the $\mathcal{EL}$ family of description logics. In: DL 2007, Bressanone, Italy. CEUR-WS, vol. 250, pp. 88–99 (2007)
11. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for Horn description logics. In: AAAI 2007, pp. 452–457 (2007)
12. Krötzsch, M., Rudolph, S., Hitzler, P.: Conjunctive queries for a tractable fragment of OWL 1.1. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007/ISWC 2007. LNCS, vol. 4825, pp. 310–323. Springer, Heidelberg (2007)
13. Lutz, C.: Inverse roles make conjunctive queries hard. In: DL 2007 (2007)
14. Lutz, C.: Two upper bounds for conjunctive query answering in $\mathcal{SHIQ}$. In: DL 2008 (2008)
15. Motik, B., Shearer, R., Horrocks, I.: Optimized reasoning in description logics using hypertableaux. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 67–83. Springer, Heidelberg (2007)
16. Ortiz, M., Šimkus, M., Eiter, T.: Worst-case optimal conjunctive query answering in description logics without inverses. In: AAAI 2008 (to appear, 2008)
17. Rosati, R.: On conjunctive query answering in $\mathcal{EL}$. In: DL 2007 (2007)
18. Rudolph, S., Krötzsch, M., Hitzler, P.: Terminological reasoning in $\mathcal{SHIQ}$ with ordered binary decision diagrams. In: AAAI 2008 (to appear, 2008)

Accommodative Belief Revision

Satu Eloranta[1], Raul Hakli[2], Olli Niinivaara[1], and Matti Nykänen[3]

[1] Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki, Finland
[2] Department of Philosophy
P.O. Box 9 (Siltavuorenpenger 20 A)
FIN-00014 University of Helsinki, Finland
firstname.lastname@cs.helsinki.fi
[3] University of Kuopio
Department of Computer Science
P.O. Box 1627
FIN-70211 Kuopio, Finland
matti.nykanen@cs.uku.fi

Abstract. Accommodative revision is a novel method of non-prioritized belief revision. The epistemic state of an agent contains both knowledge that is immune to revision and beliefs that are allowed to change. Incoming information is first revised by the knowledge of the agent, and then the epistemic state of the agent is revised using this modified input. The properties of the method are studied and examples of its use are given.

Keywords: belief change, belief revision, non-prioritized belief revision, integrity constraints, knowledge.

1 Introduction

In belief revision, an agent obtains new information about a static world. On one hand, the input may be considered as the most recent and as such the most reliable piece of information. In that case, if the new information contradicts the beliefs of the agent, it needs to give up some of the old beliefs in order to maintain consistency of beliefs [1]. However, this framework, called *prioritized belief revision*, allows even self-contradictory input to be accepted into the beliefs of the agent.

On the other hand, in *non-prioritized belief revision* (see [16] for a survey) the input is not necessarily accepted. The agent may have some information that it will refuse to give up at any situation. In computer science such information might be called *integrity constraints* [22], in philosophy *knowledge* [17]. In belief revision literature the term *core beliefs* has also been used [16].

Instead of rejecting the input that the agent knows to be impossible, we aim to find a charitable interpretation that retains as much as possible of the input. For instance, suppose that we hear that "Jaakko Kuusisto, a winner of the Sibelius violin contest, gives a concert at the forthcoming open air music festival". However, we know for a fact that although Jaakko has participated in the contest as well, it is actually his brother

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 180–191, 2008.

Pekka who has managed to win it. Our natural reaction would be to think that "the speaker must have got either the first name or the bit about the contest victory wrong. But there will be a concert by either of the two brothers, that much I can believe now".

The amount of information obtained from an "unbelievable input" may vary. Let us consider a modification of an example by Hansson [15]. Amy tells the agent that she saw a three-toed woodpecker with a red forehead and a red rump just outside her window. When it comes to birds, the agent has more knowledge than Amy: The agent knows that a three-toed woodpecker neither has a red forehead nor a red rump. The agent has various possibilities when making sense of the impossible statement. With some benevolence, it can come to one of the following conclusions: (1) Amy saw a bird with a red forehead and a red rump, but it was not a three-toed woodpecker, (2) Amy saw a bird with a red forehead and a red rump or Amy saw a three-toed woodpecker (but not one with a red forehead or a red rump), or (3) at least Amy saw some kind of a bird outside her window.

In this paper, we introduce *accommodative revision*, a method for non-prioritized belief revision. The basic idea is to use knowledge as a filter that the incoming information has to pass through before the epistemic state can be revised. The agent will modify the input to accord with its knowledge.

Our proposal has the following properties: (1) input inconsistent with knowledge will not be accepted, but the input will be modified to produce an acceptable formula prior to revising the epistemic state, (2) only knowledge is used to modify the input, and (3) the modification of the input and the revision of the epistemic state are performed as two separate phases. We will also describe an implementation that is publicly available to allow small-scale experimentation of our proposal[1].

In our two-phase revision, at first the input is revised by the knowledge of the agent, giving a new input formula. Then the epistemic state of the agent is revised by the new formula. Thus our proposal is closely related to selective revision [12] and might be considered as a generalization of the revision method presented in [2] (see Sect. 3 for comparison). Other, more recent non-prioritized belief revision proposals based on the modification of input sentence do also exist, e.g. [5], [20] and [3], but they are based on syntactic manipulation whereas our method uses previously defined (semantically-oriented) belief-revision operators already known to satisfy certain principles.

The outline of the paper is as follows. Section 2 recalls the basic ideas of belief revision. In Sect. 3 we give the definition of our method, and in Sect. 4 we study its properties. In Sect. 5 we introduce an implementation of accommodative revision with various sample operators. In Sect. 6 we analyze some examples. Section 7 is devoted for conclusions.

2 Preliminaries

In belief revision, an agent evolves its epistemic state due to incoming information called epistemic input. At first the input is classified, and the way the epistemic state will be changed depends on the result of the classification. On the meta level, the change is guarded by rationality criteria. Alchourrón, Gärdenfors and Makinson [1] have proposed a set of principles for belief revision known as the *AGM-postulates*. Darwiche

[1] http://www.cs.helsinki.fi/group/protean/abr/

and Pearl [7,8] have proposed an additional set of postulates for belief revision in order to rule out change operators that give unintuitive results in iterated revisions. The main principles in these sets of postulates are maintaining consistency of the beliefs and minimality of change.

We assume that the set of propositional formulas believed in any epistemic state is deductive closed. For each epistemic state T, let T_B denote the *belief set* of the state, that is, the deductive closure of formulas believed in the state. Let $\circ$ denote a revision operator, that is, a function from epistemic states and propositional input formulas into epistemic states. The AGM-postulates and the DP-postulates are rephrased here as follows:

(R1): $A \in (T \circ A)_B$.
(R2): If $\neg A \notin T_B$, then $(T \circ A)_B = T_B + A$.
(R3): If $A \not\models \bot$, then $\bot \notin (T \circ A)_B$.
(R4): If $A \equiv B$, then $(T \circ A)_B = (T \circ B)_B$.
(R5): $(T \circ (A \wedge B))_B \subseteq (T \circ A)_B + B$.
(R6): If $\neg B \notin (T \circ A)_B$, then $(T \circ A)_B + B \subseteq (T \circ (A \wedge B))_B$.
(DP1): If $A \models B$, then $((T \circ B) \circ A)_B = (T \circ A)_B$.
(DP2): If $A \models \neg B$, then $((T \circ B) \circ A)_B = (T \circ A)_B$.
(DP3): If $A \in (T \circ B)_B$, then $A \in ((T \circ A) \circ B)_B$.
(DP4): If $\neg A \notin (T \circ B)_B$, then $\neg A \notin ((T \circ A) \circ B)_B$.

Here A and B denote propositional (input) formulas, and $T_B + A$ denotes the deductive closure of the set $T_B \cup \{A\}$. In the AGM-postulates (R1)–(R6), the epistemic input is always prioritized over the old beliefs due to postulate (R1). Postulates (R1)–(R4) are considered basic: every (prioritized) belief-revision operator should satisfy them. Postulates (R5)–(R6) are supplementary. Note that only belief sets of epistemic states are used in the formulation: Because we do not make other assumptions about the representation of epistemic states, we do not use equality nor equivalence of epistemic states in the formulation.

If epistemic states were functionally dependent on the belief sets in the states, then the joint set of postulates would result in triviality of logic, that is, no three satisfiable but pairwise inconsistent formulas could exist [13,10]. Ruling out inconsistent epistemic states and self-contradictory epistemic input does not solve the problem [10,11]. Thus for the joint set of postulates, more elaborate epistemic states are needed.

As known [14], belief revision involves ordering among possible worlds. Those possible worlds that are minimal in the ordering are the most plausible worlds (the doxastic alternatives) in the state. The possible worlds modelling the new formula that are minimal in the ordering will be the most plausible worlds in the revised state. Spohn [24] has argued that this ordering should be part of the epistemic state, because it is altered in the process of revision.

To represent epistemic states, Spohn [24] introduced *ranking functions* (alias Ordinal Conditional Functions, OCFs), which are functions from the set of possible worlds into ordinals. The ordinal of a world is its rank. The smaller the rank, the less disbelieved is the world. The most plausible worlds (the doxastic alternatives) are the worlds with rank 0. The rank of a proposition (a set of possible worlds) is the minimum of the ranks of the worlds within it. Spohn rules out both inconsistent epistemic states

and self-contradictory epistemic input. To update the ranking, Spohn [24] also introduced a method, in which the rankings are shifted. Darwiche and Pearl [7] introduced a belief-revision operator based on Spohn's framework to accommodate the principles for iterated belief revision. We, too, shall adopt and elaborate on Spohn's framework in Sect. 5.1.

We will treat knowledge in belief change as integrity constraints have been considered in theory change. *Integrity constraints* are used to express those properties that should always hold. When defining the effect that the integrity constraints have on belief change, Katsuno and Mendelzon [18] have required that the result entails the integrity constraints. Then, with any belief-revision operator satisfying the AGM-postulates, the integrity constraints actively take part in belief revision.

Let us consider the definition by Katsuno and Mendelzon [18] in more detail. Using T to denote an epistemic state, A to denote an epistemic input, $\circ$ to denote a belief revision operator, and *IC* to denote a propositional formula expressing integrity constraints, they defined the effect of integrity constraints on belief revision as

$$T \circ^{IC} A =_{def} T \circ (A \wedge IC). \tag{1}$$

We will adopt this definition and develop it further.

3 The Principle of Accommodative Revision

Before introducing accommodative revision, let us specify our framework. For each epistemic state T, let T_B denote the belief set of the state as in Sect. 2, and let T_K denote the set of all the propositional formulas constituting the knowledge in the state. Both sets are deductively closed, thus all the tautologies will be included in them. We shall restrict our accommodative revision only to those epistemic states in which the following static constraints are satisfied:

(S1): $T_K \subseteq T_B$.
(S2): $\bot \notin T_K$.

Condition (S1) says that the agent believes what it knows, and condition (S2) says that the knowledge set does not include contradictions.

However, we do not assume that the sets T_B and T_K constitute the epistemic state. In fact, Sect. 5.1 uses Spohn's [24] ranking functions to represent epistemic states, yet the general method is independent of the representation.

We shall make a strong assumption that the set T_K can be represented by a propositional formula. This assumption can be justified when assuming that the knowledge in the epistemic state is obtained in a finite sequence of monotonous knowledge expansions starting from a state in which nothing (except tautologies) was known. Our way of meeting this assumption will be given in Sect. 5.1.

We extend Definition (1) as follows. Let T denote an epistemic state, let K denote a propositional formula representing the set T_K, and let A denote a propositional input formula. We define the *accommodative revision* of the state T by the formula A as

$$T \otimes A =_{def} T \circ (A * K) \tag{2}$$

in which $\otimes$ denotes an accommodative revision operator, $\circ$ denotes belief revision of epistemic states, and $*$ denotes revision of propositional formulas. Note that we do not propose a single operator but a scheme in which various operators can be applied.

The philosophical justification of our method is the following: We consider the modified input as an estimate of the formula that the source of the input would have given, had it had all the knowledge that our agent has.

Let us compare our proposal with some related work. As a special case of accommodative revision, we will get screened revision [19] by defining that $A * K \equiv K$ whenever $\neg A \in T_K$. Our proposal resembles the proposal by Bellot et al. [2], which also has the same two phases, but uses the same fixed distance-based revision operator to update both the input by the knowledge and the epistemic state. Our proposal lets the agent choose the two components separately, without imposing limitations on the representation of epistemic states. Thus our proposal is a generalization of theirs.

In selective revision [12], some function is to be used to replace the input by a new formula that is typically entailed by the original input. In our proposal, not only is the complement of the knowledge contracted from the input, but the knowledge is incorporated into the input. Without the latter, our proposal could be considered as an instantiation of selective revision.

4 Features of Accommodative Revision

Let us analyze some features of our accommodative revision. We want to prove that the operators in the family of accommodative revision accomplish non-prioritized belief revision satisfying the AGM postulates (R2)–(R4) and a modification of (R1). We first make some assumptions of the operators $*$ and $\circ$ used as components in accommodative revision.

We assume that the modification function $*$ is a function from pairs of propositional formulas into propositional formulas and it satisfies at least the basic AGM-postulates rephrased here for this framework as follows: (MR1): $A * K \models K$, (MR2): If $K \not\models \neg A$, then $A * K \equiv A \wedge K$, (MR3): If $K \not\models \bot$, then $A * K \not\models \bot$, (MR4): If $A \equiv A'$ and $K \equiv K'$, then $A * K \equiv A' * K'$. Here A, A', K and K' denote propositional formulas.

We assume that $\circ$ is a function from epistemic states and propositional formulas into epistemic states. We assume that the operator $\circ$ satisfies at least the basic AGM-postulates (R1)–(R4). We shall also use the following extra condition to ensure that $\circ$ does not change the knowledge in the state:

(R0): $\quad (T \circ A)_K = T_K.$

We shall restrict our accommodative revision only to those epistemic states in which the static constraints (S1) and (S2) from Sect. 3 are satisfied.

Now, using these assumptions, we can prove that an accommodative-revision operator $\otimes$ preserves these static constraints and has the following features:

(AR0): $\quad (T \otimes A)_K = T_K.$
(AR1): $\quad$ If $\neg A \notin T_K$, then $A \in (T \otimes A)_B$.
(AR2): $\quad$ If $\neg A \notin T_B$, then $(T \otimes A)_B = T_B + A$.

(AR3): $\bot \notin (T \otimes A)_B$.
(AR4): If $A \equiv A'$, then $(T \otimes A)_B = (T \otimes A')_B$.

(AR0), (AR2), and (AR4) are equivalent to postulates (R0), (R2), and (R4) correspondingly. (AR3) is stronger than (R3): It says that accommodative revision always results in epistemic states with non-contradictory belief sets. According to (AR1), the new formula is accepted, if it is not contradictory to the knowledge in the state. Note that all these are derived properties, not principles set for accommodative revision.

We can see that accommodative revision fails to satisfy the basic AGM-postulates only when the input is contradictory to the knowledge in the state: In those cases the success postulate (R1) fails. Thus accommodative revision is non-prioritized belief revision. Accommodative revision preserves the static constraints, and it is able to guarantee non-contradictory belief sets at all occasions.

Theorem 1. *If the operators $*$ and $\circ$ used as components in an accommodative revision operator $\otimes$ satisfy postulates (MR1)–(MR4) and (R0)–(R4) respectively, then given any epistemic state T satisfying constraints (S1) and (S2), then the state $T \otimes A$ satisfies (S1), (S2), and (AR0)–(AR4).*

Proof. Let T denote an epistemic state, A denote a propositional formula, and let K denote a propositional formula equivalent to T_K. We assume that $T_K \subseteq T_B$ and $\bot \notin T_K$. By definition, $(T \otimes A) = (T \circ (A * K))$.

By (R0), $(T \otimes A)_K = (T \circ (A * K))_K = T_K$, thus (AR0) holds. Then because $\bot \notin T_K$, $\bot \notin (T \otimes A)_K$ and (S2) holds. Because by (R1), $A * K \in (T \circ (A * K))_B = (T \otimes A)_B$, (MR1) gives us $A * K \models K$, thus $K \in (T \otimes A)_B$. Then by (AR0), (S1) holds.

To prove (AR1), let us assume $\neg A \notin T_K$. Then by (MR2), $A * K \equiv A \wedge K$, and (R4) gives us $(T \circ (A * K))_B = (T \circ (A \wedge K))_B$. Thus by (R1), $A \wedge K \in (T \circ (A \wedge K))_B = (T \circ (A * K))_B = (T \otimes A)_B$ and then $A \in (T \otimes A)_B$.

To prove (AR2), assume $\neg A \notin T_B$. Then by (S1), $\neg A \notin T_K$, and (MR2) gives us $A * K \equiv A \wedge K$. Then by (R4), $(T \circ (A * K))_B = (T \circ (A \wedge K))_B$. For contradiction, assume $\neg(A \wedge K) \in T_B$. Because $\neg(A \wedge K) \equiv \neg A \vee \neg K$ and $K \in T_B$, we get $\neg A \in T_B$, a contradiction. Thus $\neg(A \wedge K) \notin T_B$. Then (R2) gives us $(T \circ (A \wedge K))_B = T_B + A \wedge K$. Because $K \in T_B$, $T_B + A \wedge K = T_B + A$. Thus $(T \otimes A)_B = (T \circ (A * K))_B = T_B + A$.

Let us next prove that (AR3) holds. Because $K \not\models \bot$, (MR3) gives us $A * K \not\models \bot$. Then by (R3), $\bot \notin (T \circ (A * K))_B = (T \otimes A)_B$.

To prove (AR4), assume $A \equiv A'$. Then by (MR4), $A * K \equiv A' * K$, thus (R4) gives us $(T \otimes A)_B = (T \circ (A * K))_B = (T \circ (A' * K))_B = (T \otimes A')_B$. □

Whenever input does not contradict knowledge, accommodative revision retains the properties of the revision operator $\circ$ used in definition.

Theorem 2. *If the operators $*$ and $\circ$ satisfy postulates (MR1)–(MR4) and (R0)–(R4) respectively, then for each postulate (R5), (R6), (DP1)–(DP4), if the operator $\circ$ satisfies the postulate in question, accommodative revision also has the corresponding property below:*

(AR5): If $\neg(A \wedge B) \notin T_K$, then $(T \otimes (A \wedge B))_B \subseteq (T \otimes A)_B + B$.

(AR6): If $\neg(A \wedge B) \notin T_K$ and $\neg B \notin (T \otimes A)_B$, then $(T \otimes A)_B + B \subseteq (T \otimes (A \wedge B))_B$.
(AR7): If $A \models B$ and $\neg A \notin T_K$, then $((T \otimes B) \otimes A)_B = (T \otimes A)_B$.
(AR8): If $A \models \neg B$, $\neg A \notin T_K$ and $\neg B \notin T_K$, then $((T \otimes B) \otimes A)_B = (T \otimes A)_B$.
(AR9): If $A \in (T \otimes B)_B$, then $A \in ((T \otimes A) \otimes B)_B$.
(AR10): If $\neg A \notin (T \otimes B)_B$, then $\neg A \notin ((T \otimes A) \otimes B)_B$.

Proof. Assume $\neg(A \wedge B) \notin T_K$. By (S2), $\bot \notin T_K$, thus $\neg A \notin T_K$ and $\neg B \notin T_K$. By (MR2), $(A \wedge B) * K \equiv (A \wedge B \wedge K)$ and $A * K \equiv A \wedge K$. By (R4), $(T \otimes A \wedge B)_B = (T \circ ((A \wedge B) * K))_B = (T \circ (A \wedge B \wedge K))_B$. If $\circ$ satisfies (R5), then by (R5), (R4) and (MR2), $(T \circ (A \wedge B \wedge K))_B \subseteq (T \circ (A \wedge K))_B + B = (T \circ (A * K))_B + B = (T \otimes A)_B$, (AR5) holds. To prove (AR6), assume also that $\neg B \notin (T \otimes A)_B$. Then by definition, $\neg B \notin (T \circ (A \wedge K))_B$. If $\circ$ satisfies (R6), then by (R6), (R4) and (MR2), $(T \otimes A)_B + B = (T \circ (A * K))_B + B = (T \circ (A \wedge K))_B + B \subseteq (T \circ (A \wedge B \wedge K))_B = (T \circ ((A \wedge B) * K))_B = (T \otimes (A \wedge B))_B$, (AR6) holds.

To prove (AR7), assume $\neg A \notin T_K$ and $A \models B$. Then $\neg B \notin T_K$ and $(A \wedge K) \models (B \wedge K)$. If $\circ$ satisfies (DP1), then by (MR2), (R4), and (DP1), $((T \otimes B) \otimes A)_B = ((T \circ (B * K)) \circ (A * K))_B = ((T \circ (B \wedge K)) \circ (A \wedge K))_B = (T \circ (A \wedge K))_B = (T \circ (A * K))_B = (T \otimes A)_B$.

To prove (AR8), assume $\neg A, \neg B \notin T_K$ and $A \models \neg B$. Thus $A \wedge K \models \neg B \wedge K$. If $\circ$ satisfies (DP2), then by (MR2), (R4), and (DP2), $((T \otimes B) \otimes A)_B = ((T \circ (B * K)) \circ (A * K))_B = ((T \circ (B \wedge K)) \circ (A \wedge K))_B = (T \circ (A \wedge K))_B = (T \circ (A * K))_B = (T \otimes A)_B$.

To prove (AR9), let us assume $A \in (T \otimes B)_B$. Then by (AR3), (S1), and (AR0), $\neg A \notin T_K$ and $A \wedge K \in (T \otimes B)_B = (T \circ (B * K))_B$. By (MR2) and (R4) $((T \otimes A) \otimes B)_B = ((T \circ (A * K)) \circ (B * K))_B = ((T \circ (A \wedge K)) \circ (B * K))_B$. If $\circ$ satisfies (DP3), then by (DP3), $A \wedge K \in ((T \circ (A \wedge K)) \circ (B * K))_B = ((T \circ (A * K)) \circ (B * K))_B = ((T \otimes A) \otimes B)_B$.

To prove (AR10), assume $\neg A \notin (T \otimes B)_B$. Then by (S1) and (AR0), $\neg A \notin (T \otimes B)_K = T_K$, and thus by (MR2), $(A * K) \equiv (A \wedge K)$. For contradiction, assume $\neg(A \wedge K) \in (T \otimes B)_B$. Because $\neg(A \wedge K) \equiv \neg A \vee \neg K$ and $K \in (T \otimes B)_B$, we get $\neg A \in (T \otimes B)_B$, a contradiction. Thus $\neg(A \wedge K) \notin (T \otimes B)_B = (T \circ (B * K))_B$. If $\circ$ satisfies (DP4), then by (DP4) and (R4), $\neg(A \wedge K) \notin ((T \circ (A \wedge K)) \circ (B * K))_B = ((T \circ (A * K)) \circ (B * K))_B = ((T \otimes A) \otimes B)_B$. □

5 An Implementation

5.1 The Underlying Framework

As stated in Sect. 3, we have adopted Spohn's ranking functions (OCFs) κ [24, Definition 4] as our representation for the epistemic state T. However, we restrict their range to the natural numbers augmented with infinity ∞. Intuitively, the *rank* $\kappa(w)$ is the agent's degree of disbelief towards this world w being the actual one. This ranking extends to formulas A as the minimum rank of their models: $\kappa(A) = \min\{\kappa(w) \colon w \in \mathrm{Mod}(A)\}$. Their connection to beliefs is that $T_B = \{A \colon \kappa(\neg A) > 0\}$ and to knowledge that $T_K = \{A \colon \kappa(\neg A) = \infty\}$. The OCF definition has two more requirements: (i) $\kappa^{-1}(0) \neq \emptyset$ so

that the lowest rank is normalized to be 0. (ii) The agent must be indifferent towards new vocabulary: if $\kappa(w) \neq \kappa(w')$ then w and w' must disagree on some atomic formula p which it has already encountered in some formula.

At least one of $\kappa(A)$ or $\kappa(\neg A)$ is 0 [24, Theorem 2 (a)]. If they both are, then the agent believes neither A nor $\neg A$. For instance, in the *initial* ranking function κ_0, which is everywhere 0, only the tautologies are known and nothing else is believed.

Spohn's A, α-*conditionalization* [24, Definition 6] constructs from a given OCF κ another OCF

$$\kappa_{A,\alpha}(w) =_{def} \begin{cases} \kappa(w) - \kappa(A) & \text{if } \kappa(w) < \infty \text{ and } w \in \mathrm{Mod}(A) \\ \alpha + (\kappa(w) - \kappa(\neg A)) & \text{if } \kappa(w) < \infty \text{ and } w \notin \mathrm{Mod}(A) \\ \kappa(w) & \text{if } \kappa(w) = \infty \end{cases} \tag{3}$$

where $\mathrm{Mod}(A)$ is ranked to 0 while $\mathrm{Mod}(\neg A)$ is ranked to the given constant $\alpha > 0$ without altering the distances within these two moving parts. However, we must ensure $\kappa(A) < \infty$ to meet requirement (i). Following Darwiche and Pearl [7], we use $\kappa_{A,1}$ as our belief revision operation when A is not believed and κ otherwise.

5.2 The Program

We have implemented the approach taken in Sect. 5.1 as a library of functions in the functional programming language Haskell [21]. This library can be loaded into a Haskell interpreter such as GHCi (see `http://haskell.org/ghc/`) which then provides a text-based environment where the user can experiment with different instantiations of our operator scheme. It offers the following three functions:

$$\texttt{initial} = \kappa_0. \tag{4}$$

$$\texttt{know}\ \ \kappa\ \ A = \begin{cases} \kappa_{A,\infty} & \text{if } \kappa(A) < \infty \\ \kappa & \text{otherwise.} \end{cases} \tag{5}$$

$$\texttt{hear}_*\ \ \kappa\ \ A = \begin{cases} \kappa & \text{if } \kappa(A) = 0 \\ \kappa_{A,1} & \text{if } 0 < \kappa(A) < \infty \\ \texttt{hear}_*\ \ \kappa\ \ (A * K) & \text{otherwise.} \end{cases} \tag{6}$$

Constant (4) is the initial OCF. Requirement (ii) shows how the logical vocabulary can be extended dynamically as needed, so we do not have to give it explicitly. Function (5) expands the knowledge at κ with A, but discards A if $\neg A \in T_K$ already. Function (6) is our revision scheme from Definition (2). Here the higher-order parameter $*$ is the syntactic propositional formula revision operator used. This representation of T_K as a single formula K assumed in Sect. 3 is now justified since the current T_K has developed from the `initial` OCF through a finite sequence of `know` steps.

The main goal in designing the implementation was to allow experimenting with different $*$. It is namely not clear at the outset which one corresponds best to our intuition about "the closest alternative to A which I can believe", as witnessed by the different choices in the woodpecker example in the introduction. Currently the implementation offers the four operators described in Sect. 5.3. Defining additional ones is also straightforward using Haskell.

Compared with the COBA 2.0 system [9], ours is decidedly much narrower in scope: it has neither a graphical user interface (GUI) nor a satisfiability (SAT) solver to support processing any but the smallest knowledge bases. On the other hand, experimentation dictates that our implementation must in turn be able to retain and view several versions for the same knowledge base concurrently in memory, namely the revisions of the same base using different operators $*$. This is conveniently supplied by the underlying Haskell interpreter.

5.3 Modification Operators

We shall next give the definitions of the four semantically-oriented belief-revision operators we implemented as modification operators.

For defining these operators, we define the *difference* $w \triangle w'$ between two worlds $w, w' \in \mathcal{W}$ as the set of the atomic formulas having a different truth value in w and w', that is, $w \triangle w' = (w \setminus w') \cup (w' \setminus w)$. These sets are compared either by using the subset relation or the cardinalities of the sets:

$$\begin{aligned}
\mathit{diff}(T, A) &= \min(\{w \triangle w' : w \in \mathrm{Mod}(T), w' \in \mathrm{Mod}(A)\}, \subseteq),\\
\mathit{dist}(T, A) &= \min(\{|w \triangle w'| : w \in \mathrm{Mod}(T), w' \in \mathrm{Mod}(A)\}, \leq),\\
\mathit{p_diff}(w, A) &= \min(\{w \triangle w' : w' \in \mathrm{Mod}(A)\}, \subseteq).
\end{aligned}$$

When determining the minimal difference, *diff* and *p_diff* use the subset relation in comparison, while *dist* compares the cardinalities of the sets. The first two of the functions search for the minimal differences between two model sets, while the last function compares one model to a set of models pointwise.

We use four semantically-oriented belief-revision operators as modification operators: Dalal's [6] operator $*_D$, Satoh's [23] operator $*_S$, Weber's [25] operator $*_W$, and Borgida's [4] operator $*_B$. For all the operators, we define $\mathrm{Mod}(A * K) = \mathrm{Mod}(K)$ whenever $\mathrm{Mod}(A) = \emptyset$, otherwise the operators are defined as follows [18]:

$$\begin{aligned}
\mathrm{Mod}(A *_D K) &= \{w \in \mathrm{Mod}(K) : \exists w' \in \mathrm{Mod}(A), |w \triangle w'| = \mathit{dist}(A, K)\},\\
\mathrm{Mod}(A *_S K) &= \{w \in \mathrm{Mod}(K) : \exists w' \in \mathrm{Mod}(A), w \triangle w' \in \mathit{diff}(A, K)\},\\
\mathrm{Mod}(A *_W K) &= \{w \in \mathrm{Mod}(K) : \exists w' \in \mathrm{Mod}(T), w \triangle w' \subseteq \bigcup \mathit{diff}(A, K)\},\\
\mathrm{Mod}(A *_B K) &= \mathrm{Mod}(A \wedge K), \text{if } A \wedge K \text{ is satisfiable, otherwise}\\
\mathrm{Mod}(A *_B K) &= \textstyle\bigcup_{w \in \mathrm{Mod}(A)} \{w' \in \mathrm{Mod}(K) : w \triangle w' \in \mathit{p_diff}(w, K)\}.
\end{aligned}$$

Only Dalal's operator satisfies all (R1)–(R6), Satoh's and Borgida's operators satisfy (R1)–(R5), Weber's operator satisfies (R1)–(R4).

The operators define rules to produce the new set of models, but they do not define the outcome of the addition as a formula. A formula A' may be the result of the revision $A * K$, if $\mathrm{Mod}(A') = \mathrm{Mod}(A * K)$.

6 Experiments on Accommodative Revision

We shall next experiment. Let us start by considering the classic, simple example of a dinosaur and a vase given by Fermé and Hansson [12].

Example 1 ("Dinosaur broke grandma's vase!")
On your return home, your son tells you that a dinosaur has broken grandmother's vase in the living room. Assuming that you know that dinosaurs do not exist this claim cannot be entirely correct, but you may still want to accept as true that the vase has been broken[2]. We can formalize the example as follows:

a grandma's vase is intact,
b grandma's vase is in the living room,
c a dinosaur broke grandma's vase,
d dinosaurs exist.

What you know is $\neg d \wedge (\neg d \rightarrow \neg c)$. Assume that you have come to believe (independently) that a and that b. This means that we have four epistemically possible worlds with both c and d false in each. The most plausible one is the world in which a and b are true. Thus prior to the revision the epistemic state is

world	a	b	c	d	rank
w_{12}	1	1	0	0	0
w_4	0	1	0	0	1
w_8	1	0	0	0	1
w_0	0	0	0	0	2

Your son then tells you that $b \wedge c \wedge \neg a$. Revising this with a formula K representing your knowledge, $\neg c \wedge \neg d$, gives as result $A * K \equiv \neg a \wedge b \wedge \neg c \wedge \neg d$.

Revision of beliefs with this formula makes the only world satisfying this formula to become the most plausible one. As a result, you believe that grandma's vase is broken in the living room but not that it was a dinosaur that broke it.

The execution of the example with our Haskell implementation confirms the result. The example is so simple that all the modification operators we implemented result in the same revised epistemic state:

world	a	b	c	d	rank
w_4	0	1	0	0	0
w_{12}	1	1	0	0	1
w_8	1	0	0	0	2
w_0	0	0	0	0	3

As an example of a case where different modification functions give us different results, let us consider again the three-toed woodpecker example from the introduction.

Example 2 (The three-toed woodpecker)
We formalize the situation as follows:

a Amy saw a bird,
b Amy saw a three-toed woodpecker,
c Amy saw a bird with a red forehead,
d Amy saw a bird with a red rump.

[2] In fact, the world changes in this example so it might be more appropriate to do a belief update rather than a revision, but we will ignore this now since the example has often been used to illustrate non-prioritized revision.

Initially the agent knows that a three-toed woodpecker neither has a red forehead nor a red rump, and that if Amy saw a three-toed woodpecker or she saw a bird with a red forehead or she saw a bird with a red rump outside her window, then she saw a bird outside her window. Thus the agent has the knowledge $K \equiv (b \rightarrow (\neg c \wedge \neg d)) \wedge ((b \vee c \vee d) \rightarrow a)$.

Amy then tells that $a \wedge b \wedge c \wedge d$. Now, the agent may have different results of the modification of the input depending on which modification function is used:

$$(a \wedge b \wedge c \wedge d) *_D K \equiv (a \wedge \neg b \wedge c \wedge d),$$
$$(a \wedge b \wedge c \wedge d) *_S K \equiv (a \wedge \neg b \wedge c \wedge d) \vee (a \wedge b \wedge \neg c \wedge \neg d),$$
$$(a \wedge b \wedge c \wedge d) *_B K \equiv (a \wedge \neg b \wedge c \wedge d) \vee (a \wedge b \wedge \neg c \wedge \neg d),$$
$$(a \wedge b \wedge c \wedge d) *_W K \equiv (a \wedge \neg b) \vee (a \wedge b \wedge \neg c \wedge \neg d).$$

Here Dalal's operator gave the result (1) mentioned in the example in Sect. 1, Satoh's and Borgida's operators gave the result (2), and Weber's operator gave the result (3).

However, had Amy told that $b \wedge c \wedge d$, the results would have been different:

$$(b \wedge c \wedge d) *_D K \equiv (a \wedge \neg b \wedge c \wedge d),$$
$$(b \wedge c \wedge d) *_S K \equiv (a \wedge \neg b \wedge c \wedge d) \vee (a \wedge b \wedge \neg c \wedge \neg d),$$
$$(b \wedge c \wedge d) *_B K \equiv (a \wedge \neg b \wedge c \wedge d) \vee (a \wedge b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge \neg c \wedge \neg d)$$
$$(b \wedge c \wedge d) *_W K \equiv K.$$

Here Dalal's and Satoh's operators still give charitable results, but Borgida's and Weber's operators do not.

In general, the more permissive is the revision operator, the less information is left to be obtained from the modified input (but the less likely it is that the agent has ruled out the actual state of affairs). Also the combination of pointwise revision and incomplete input is likely to give uncharitable results.

7 Conclusion

To illustrate the effect of knowledge on belief revision, we have presented a non-prioritized belief revision method that we call accommodative revision. In this method, the input is first revised with the knowledge of the agent. Beliefs are then revised with the resulting modified input. The properties of the method have been studied, a prototype implementation has been described, and some experiments have been analyzed.

The two components of accommodative revision can be chosen separately. The method does not call for any particular representation of epistemic states nor any principles on the components of accommodative revision other than the basic AGM-postulates. Accommodative revision is not meant to be the only way for an agent to update its epistemic state. Rather, it is to provide the agent with a set of tools to build convenient new belief-change operators to go with the old ones.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. The Journal of Symbolic Logic 50(2), 510–530 (1985)
2. Bellot, D., Godefroid, C., Han, P., Prost, J.P., Schlechta, K., Wurbel, E.: A semantical approach to the concept of screened revision. Theoria 63, 24–33 (1997)

3. Benferhat, S., Kaci, S., Le Berre, D., Williams, M.-A.: Weakening conflicting information for iterated revision and knowledge integration. Artificial Intelligence 153, 339–371 (2004)
4. Borgida, A.: Language features for flexible handling of exceptions in information systems. ACM Transactions on Database Systems 10(4), 565–603 (1985)
5. Cravo, M.R., Cachopo, J.P., Cachopo, A.C., Martins, J.P.: Permissive belief revision. In: Brazdil, P.B., Jorge, A.M. (eds.) EPIA 2001. LNCS (LNAI), vol. 2258, pp. 335–348. Springer, Heidelberg (2001)
6. Dalal, M.: Investigations into a theory of knowledge base revision: Preliminary report. In: Proceedings of the AAAI-88 Seventh National Conference on Artificial Intelligence, pp. 475–479. Morgan Kaufmann, San Francisco (1988)
7. Darwiche, A., Pearl, J.: On the logic of iterated belief revision. In: Fagin, R. (ed.) Theoretical Aspects of Reasoning about Knowledge. Proceedings of the Fifth Conference (TARK 1994), pp. 5–23 (1994)
8. Darwiche, A., Pearl, J.: On the logic of iterated belief revision. Artificial Intelligence 89(1-2), 1–29 (1997)
9. Delgrande, J.P., Liu, D.H., Schaub, T., Thiele, S.: COBA 2.0: A consistency-based belief change system. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 78–90. Springer, Heidelberg (2007)
10. Eloranta, S.: Updating the knowledge base (in Finnish). Technical Report C-1995-5, Department of Computer Science, University of Helsinki (1995)
11. Eloranta, S.: Dynamic features of the knowledge base. Technical Report C-2004-90, Department of Computer Science, University of Helsinki (2004)
12. Fermé, E.L., Hansson, S.O.: Selective revision. Studia Logica 63(3), 331–342 (1999)
13. Freund, M., Lehmann, D.: Belief revision and rational inference. Technical Report TR 94-16, Hebrew University (1994)
14. Grove, A.: Two modellings for theory change. Journal of Philosophical Logic 17, 157–170 (1988)
15. Hansson, S.O.: Belief contraction without recovery. Studia Logica 50(2), 251–260 (1991)
16. Hansson, S.O.: A survey of non-prioritized belief revision. Erkenntnis 50(2-3), 413–427 (1999)
17. Hintikka, J.: Knowledge and Belief. Cornell University Press (1962)
18. Katsuno, H., Mendelzon, A.O.: Propositional knowledgebase revision and minimal change. Artificial Intelligence 52(3), 263–294 (1992)
19. Makinson, D.: Screened revision. Theoria 63, 14–23 (1997)
20. Maranhão, J.: Refinement: A tool to deal with inconsistencies. In: Proceedings of the Eighth International Conference on Artificial Intelligence and Law, ICAIL 2001, pp. 52–59 (2001)
21. Jones, S.P.: Haskell 98 Language and Libraries: the Revised Report. Cambridge University Press, Cambridge (2003); Journal of Functional Programming 13(1) (2003)
22. Reiter, R.: On integrity constraints. In: 2nd Conference on Theoretical Aspects of Reasoning about Knowledge, pp. 97–111. Morgan Kaufmann, San Francisco (1988)
23. Satoh, K.: Nonmonotonic reasoning by minimal belief revision. In: Proceedings of the International Conference on Fifth Generation Computer Systems, pp. 455–462 (1988)
24. Spohn, W.: Ordinal conditional functions: a dynamic theory of epistemic state. In: Harper, W.L., Skyrms, B. (eds.) Causation in Decision, Belief Change, and Statistics, vol. II, pp. 105–134. Kluwer Academic Publishers, Dordrecht (1988)
25. Weber, A.: Updating propositional formulas. In: Proceedings of the First International Conference on Expert Database Systems (1986)

Reasoning about Typicality in Preferential Description Logics

Laura Giordano[1], Valentina Gliozzi[2], Nicola Olivetti[3], and Gian Luca Pozzato[2]

[1] Dip. di Informatica - Univ. Piemonte O. "A. Avogadro"
laura@mfn.unipmn.it
[2] Dip. di Informatica, Università di Torino
{gliozzi,pozzato}@di.unito.it
[3] LSIS-UMR CNRS 6168 Univ. "P. Cézanne"
nicola.olivetti@univ-cezanne.fr

Abstract. In this paper we propose a nonmonotonic extension $\mathcal{ALC} + \mathbf{T}_{min}$ of the Description Logic $\mathcal{ALC}$ for reasoning about prototypical properties and inheritance with exception. The logic $\mathcal{ALC} + \mathbf{T}_{min}$ is built upon a previously introduced (monotonic) logic $\mathcal{ALC} + \mathbf{T}$, that is obtained by adding a typicality operator $\mathbf{T}$ to $\mathcal{ALC}$. The operator $\mathbf{T}$ is intended to select the "most normal" or "most typical" instances of a concept, so that knowledge bases may contain subsumption relations of the form"$\mathbf{T}(C)$ is subsumed by P", expressing that typical C-members have the property P. In order to perform nonmonotonic inferences, we define a "minimal model" semantics $\mathcal{ALC} + \mathbf{T}_{min}$ for $\mathcal{ALC} + \mathbf{T}$. The intuition is that preferred, or minimal models are those that maximise typical instances of concepts. By means of $\mathcal{ALC} + \mathbf{T}_{min}$ we are able to infer defeasible properties of (explicit or implicit) individuals. We also present a tableau calculus for deciding $\mathcal{ALC} + \mathbf{T}_{min}$ entailment.

1 Introduction

The family of description logics (DLs) is one of the most important formalisms of knowledge representation. They have a well-defined semantics based on first-order logic and offer a good trade-off between expressivity and complexity. DLs have been successfully implemented by a range of systems and they are at the base of languages for the semantic web such as OWL.

A DL knowledge base (KB) comprises two components: the TBox, containing the definition of concepts (and possibly roles), and a specification of inclusions relations among them, and the ABox containing instances of concepts and roles. Since the very objective of the TBox is to build a taxonomy of concepts, the need of representing prototypical properties and of reasoning about defeasible inheritance of such properties naturally arises. The traditional approach is to handle defeasible inheritance by integrating some kind of nonmonotonic reasoning mechanism. This has led to study nonmonotonic extensions of DLs [1,2,3,5,6,12]. However, finding a suitable nonmonotonic extension for inheritance with exceptions is far from obvious.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 192–205, 2008.

To give a brief account, [1] proposes the extension of DL with Reiter's default logic. However, the same authors have pointed out that this integration may lead to both semantical and computational difficulties. Furthermore, Reiter's default logic does not provide a direct way of modeling inheritance with exceptions. This has motivated the study of extensions of DLs with prioritized defaults [12,2]. A more general approach is undertaken in [5], where an extension of DL is proposed with two epistemic operators. This extension, called $\mathcal{ALCK}_{\mathcal{NF}}$, allows one to encode Reiter's default logic as well as to express epistemic concepts and procedural rules. However, this extension has a rather complicated modal semantics, so that the integration with the existing systems requires significant changes to the standard semantics of DLs. [9] extends the work in [5] by providing a translation of an $\mathcal{ALCK}_{\mathcal{NF}}$ KB to an equivalent *flat* KB and by defining a simplified tableau algorithm for flat KBs, which includes an optimized minimality check. In [3] an extension of DL with circumscription is proposed to express prototypical properties with exceptions, by introducing "abnormality" predicates whose extension is minimized. The authors provide algorithms for checking satisfiability, subsumption and instance checking which are proved to have an optimal complexity, but are based on massive nondeterministic guessing. A calculus for circumscription in DL has not been developed yet. Moreover, the use of circumscription to model inheritance with exceptions is not that straightforward.

In this work, we propose a new nonmonotonic logic $\mathcal{ALC} + \mathbf{T}_{min}$ for defeasible reasoning in description logic. Our starting point is the monotonic logic $\mathcal{ALC} + \mathbf{T}$ introduced in [7], obtained by adding a typicality operator $\mathbf{T}$ to $\mathcal{ALC}$. The intended meaning of the operator $\mathbf{T}$, for any concept C, is that $\mathbf{T}(C)$ singles out the instances of C that are considered as "typical" or "normal". Thus assertions as "normally students do not pay taxes", or "typically users do not have access to confidential files" [3] are represented by $\mathbf{T}(\mathit{Student}) \sqsubseteq \neg \mathit{TaxPayer}$ and $\mathbf{T}(\mathit{User}) \sqsubseteq \neg\exists \mathit{HasAccess}.\mathit{ConfidentialFile}$. As shown in [7], the operator $\mathbf{T}$ is characterised by a set of postulates that are essentially a reformulation of KLM [10] axioms of preferential logic $\mathbf{P}$, namely the assertion $\mathbf{T}(C) \sqsubseteq P$ is equivalent to the conditional assertion $C \mathrel{|\!\sim} P$ of $\mathbf{P}$. It turns out that the semantics of the typicality operator can be equivalently specified by a suitable modal logic.

The idea underlying the modal interpretation is that there is a global preference relation (a strict partial order) $<$ on individuals, so that typical instances of a concept C can be defined as the instances of C that are minimal with respect to $<$. In this modal logic, $<$ works as an accessibility relation R with $R(x, y) \equiv y < x$, so that we can define $\mathbf{T}(C)$ as $C \sqcap \Box \neg C$. The preference relation $<$ does not have infinite descending chains as we adopt the so-called Smoothness condition or Limit Assumption of conditional logics. As a consequence, the corresponding modal operator $\Box$ has the same properties as in Gödel-Löb modal logic G of arithmetic provability.

In our setting, we assume that a KB comprises, in addition to the standard TBox and ABox, a set of assertions of the type $\mathbf{T}(C) \sqsubseteq D$ where D is a concept not mentioning $\mathbf{T}$. For instance, let the KB contain: $\mathbf{T}(\mathit{Student}) \sqsubseteq \neg \mathit{TaxPayer}$;

$\mathbf{T}(Student \sqcap Worker) \sqsubseteq TaxPayer$; $\mathbf{T}(Student \sqcap Worker \sqcap \exists HasChild.\top) \sqsubseteq \neg TaxPayer$, corresponding to the assertions: normally a student does not pay taxes, normally a working student pays taxes, but normally a working student having children does not pay taxes. Suppose further that the ABox contains alternatively the following facts about $john$: 1. $Student(john)$; 2. $Student(john)$, $Worker(john)$; 3. $Student(john)$, $Worker(john)$, $\exists HasChild.\top(john)$. We would like to infer the expected (defeasible) conclusion about $john$ in each case: 1. $\neg TaxPayer(john)$, 2. $TaxPayer(john)$, 3. $\neg TaxPayer(john)$. Moreover, we would like to infer (defeasible) properties also of individuals implicitly introduced by existential restrictions, for instance, if the ABox further contains $\exists HasChild.(Student \sqcap Worker)(jack)$ it should derive (defeasibly) the "right" conclusion $\exists HasChild.TaxPayer(jack)$ in the latter. Finally, adding irrelevant information should not affect the conclusions. Given the KB as above, one should be able to infer as well $\mathbf{T}(Student \sqcap SportLover) \sqsubseteq \neg TaxPayer$ and $\mathbf{T}(Student \sqcap Worker \sqcap SportLover) \sqsubseteq TaxPayer$, as $SportLover$ is irrelevant with respect to being a TaxPayer or not. For the same reason, the conclusion about $john$ being a TaxPayer or not should not be influenced by adding $SportLover(john)$ to the ABox.

The monotonic logic $\mathcal{ALC}+\mathbf{T}$ is not sufficient to perform the kind of defeasible reasoning illustrated above. Concerning the example, we get for instance that: KB $\cup$ $\{Student(john), Worker(john)\} \not\models TaxPayer(john)$; KB $\not\models \mathbf{T}(Student \sqcap SportLover) \sqsubseteq \neg TaxPayer$. In order to derive the conclusion about $john$ we should know (or assume) that $john$ is a typical working student, but we do not dispose of this information. Similarly, in order to derive that also a typical student who loves sport must not pay taxes, we must be able to infer or assume that a "typical student loving sport" is also a "typical student", since there is no reason why it should not be the case; this cannot be derived by the logic itself given the nonmonotonic nature of $\mathbf{T}$. The basic monotonic logic $\mathcal{ALC} + \mathbf{T}$ is then too weak to enforce these extra assumptions, so that we need an additional mechanism to perform defeasible inferences. In [7], we proposed a completion of the KB that adds, for each individual, the assumption that the individual is a typical member of the *most specific concept* to which it belongs. However, this solution presents some difficulties: (i) it is not clear how to take into account implicit individuals, (ii) the completion might be inconsistent, so that we must consider alternative maximal completions, (iii) it is not clear whether and how the completion has to take into account concept instances that are inferred from previous typicality assumptions introduced by the completion itself (this would require a kind of fixpoint definition).

In this work we follow another approach, rather than defining an ad-hoc mechanism to perform defeasible inferences or making nonmonotonic assumptions, we strenghten the semantics of the logic by proposing a minimal model semantics. Intuitively, the idea is to restrict our consideration to models that maximise typical instances of a concept. In order to define the preference relation on models we take advantage of the modal semantics of $\mathcal{ALC} + \mathbf{T}$: the preference relation on models (with the same domain) is defined by comparing, for

each individual, the set of modal (or more precisely $\Box$-ed) concepts containing the individual in the two models. Similarly to circumscription, where we must specify a set of minimised predicates, here we must specify a set of concepts $\mathcal{L}_T$ of which we want to maximise the set of typical instances (it may just be the set of all concepts occurring in the knowledge base). We call the new logic $\mathcal{ALC} + \mathbf{T}_{min}$ and we denote by $\models_{min}^{\mathcal{L}_T}$ semantic entailment determined by minimal models. Taking the KB of the examples above we obtain, for instance, KB $\cup$ $\{Student(john), Worker(john)\} \models_{min}^{\mathcal{L}_T} TaxPayer(john)$; KB $\cup$ $\{\exists HasChild.(Student \sqcap Worker)(jack)\}$ $\models_{min}^{\mathcal{L}_T}$ $\exists HasChild.TaxPayer(jack)$ and KB $\models_{min}^{\mathcal{L}_T}$ $\mathbf{T}(Student \sqcap SportLover) \sqsubseteq \neg TaxPayer$. As the second example shows, we are able to infer the intended conclusion also for the implicit individuals.

We provide a decision procedure for checking satisfiability and validity in $\mathcal{ALC} + \mathbf{T}_{min}$. Our decision procedure has the form of tableaux calculus, with a two-step tableau construction. The idea is that the top level construction generates open branches that are candidates to represent minimal models, whereas the auxiliary construction checks whether a candidate branch represents indeed a minimal model. Termination is ensured by means of a standard blocking mechanism. Our procedure can be used to determine constructively an upper bound of the complexity of $\mathcal{ALC} + \mathbf{T}_{min}$. Namely we obtain that checking query entailment for $\mathcal{ALC} + \mathbf{T}_{min}$ is in CO-NEXP$^{\mathrm{NP}}$.

2 The Logic $\mathcal{ALC} + \mathbf{T}$

In this section, we summarize the characteristics of the original $\mathcal{ALC}+\mathbf{T}$, which is an extension of $\mathcal{ALC}$ by a typicality operator $\mathbf{T}$. Given an alphabet of concept names $\mathcal{C}$, of role names $\mathcal{R}$, and of individuals $\mathcal{O}$, the language $\mathcal{L}$ of the logic $\mathcal{ALC} + \mathbf{T}$ is defined by distinguishing *concepts* and *extended concepts* as follows: (Concepts) $A \in \mathcal{C}$ and $\top$ are *concepts* of $\mathcal{L}$; if $C, D \in \mathcal{L}$ and $R \in \mathcal{R}$, then $C \sqcap D, C \sqcup D, \neg C, \forall R.C, \exists R.C$ are *concepts* of $\mathcal{L}$. (Extended concepts) if C is a concept of $\mathcal{L}$, then C and $\mathbf{T}(C)$ are *extended concepts* of $\mathcal{L}$, and all the boolean combinations of extended concepts are extended concepts of $\mathcal{L}$. A knowledge base is a pair (TBox, ABox). TBox contains subsumptions $C \sqsubseteq D$, where $C \in \mathcal{L}$ is either a concept or an extended concept $\mathbf{T}(C')$, and $D \in \mathcal{L}$ is a concept. ABox contains expressions of the form $C(a)$ and aRb where $C \in \mathcal{L}$ is an extended concept, $R \in \mathcal{R}$, and $a, b \in \mathcal{O}$.

Definition 1 (Semantics of $\mathcal{ALC}+\mathbf{T}$). *A model $\mathcal{M}$ is any structure $\langle \Delta, <, I \rangle$, where Δ is the domain; $<$ is a strict partial order over Δ. For all $S \subseteq \Delta$, we define $Min_<(S) = \{a : a \in S$ and $\nexists b \in S$ s.t. $b < a\}$. We say that $<$ satisfies the* Smoothness Condition *i.e., for all $S \subseteq \Delta$, for all $a \in S$, either $a \in Min_<(S)$ or $\exists b \in Min_<(S)$ such that $b < a$. I is the extension function that maps each extended concept C to $C^I \subseteq \Delta$, and each role R to a $R^I \subseteq \Delta^I \times \Delta^I$. For concepts (built from operators of $\mathcal{ALC}$), C^I is defined in the usual way. For the $\mathbf{T}$ operator: $(\mathbf{T}(C))^I = Min_<(C^I)$. A model satisfying a Knowledge Base (TBox,ABox) is defined as usual. We assume the unique name assumption.*

Notice that the meaning of $\mathbf{T}$ can be split into two parts: for any a of the domain Δ, $a \in (\mathbf{T}(C))^I$ just in case (i) $a \in C^I$, and (ii) there is no $b \in C^I$ such that $b < a$. In order to isolate the second part of the meaning of $\mathbf{T}$ (for the purpose of the calculus that we will present later), we introduce a new modality $\Box$. The basic idea is simply to interpret the preference relation $<$ as an accessibility relation. By the Smoothness Condition, it turns out that $\Box$ has the properties as in Gödel-Löb modal logic of provability G. The Smoothness Condition ensures that typical elements of C^I exist whenever $C^I \neq \emptyset$, by preventing infinitely descending chains of elements. This condition therefore corresponds to the finite-chain condition on the accessibility relation (as in G). The interpretation of $\Box$ in $\mathcal{M}$ is as follows:

Definition 2. $(\Box C)^I = \{a \in \Delta \mid \textit{for every } b \in \Delta, \textit{ if } b < a \textit{ then } b \in C^I\}$

We have that a is a typical instance of C ($a \in (\mathbf{T}(C))^I$) iff $a \in (C \sqcap \Box\neg C)^I$. Since we only use $\Box$ to capture the meaning of $\mathbf{T}$, in the following we will always use the modality $\Box$ followed by a negated concept, as in $\Box\neg C$.

3 The Logic $\mathcal{ALC} + \mathbf{T}_{min}$

The logic $\mathcal{ALC} + \mathbf{T}$ allows one to reason about typicality. As a difference with respect to standard $\mathcal{ALC}$, in $\mathcal{ALC} + \mathbf{T}$ we can consistently express, for instance, the fact that three different concepts, as *student*, *working student* and *working student with children*, have a different status as taxpayers. As we have seen in the introduction, this can be consistently expressed by including in a knowledge base the three formulas: $\mathbf{T}(\mathit{Student}) \sqsubseteq \neg \mathit{TaxPayer}$; $\mathbf{T}(\mathit{Student} \sqcap \mathit{Worker}) \sqsubseteq \mathit{TaxPayer}$; $\mathbf{T}(\mathit{Student} \sqcap \mathit{Worker} \sqcap \exists \mathit{HasChild}.\top) \sqsubseteq \neg \mathit{TaxPayer}$. Assume that *john* is an instance of the concept $\mathit{Student} \sqcap \mathit{Worker} \sqcap \exists \mathit{HasChild}.\top$. What can we conclude about *john*? If the ABox contains the assertion $(*)$ $\mathbf{T}(\mathit{Student} \sqcap \mathit{Worker} \sqcap \exists \mathit{HasChild}.\top)(\mathit{john})$, then, in $\mathcal{ALC} + \mathbf{T}$, we can conclude that $\neg \mathit{TaxPayer}(\mathit{john})$. However, in the absence of (*), we cannot derive $\neg \mathit{TaxPayer}(\mathit{john})$.

We would like to infer that individuals are typical instances of the concepts they belong to, if consistent with the KB. In order to maximize the typicality of instances, we define a preference relation on models, and we introduce a semantic entailment determined by minimal models. Informally, we prefer a model $\mathcal{M}$ to a model $\mathcal{N}$ if $\mathcal{M}$ contains more typical instances of concepts than $\mathcal{N}$.

Given a KB, we consider a finite set $\mathcal{L}_T$ of concepts occurring in the KB, the typicality of whose instances we want to maximize. The maximization of the set of typical instances will apply to individuals explicitly occurring in the ABox as well as to implicit individuals. We assume that the set $\mathcal{L}_T$ contains at least all concepts C such that $\mathbf{T}(C)$ occurs in the KB.

We have seen that a is a typical instance of a concept C ($a \in (\mathbf{T}(C))^I$) when it is an instance of C and there is not another instance of C preferred to a, i.e. $a \in (C \sqcap \Box\neg C)^I$. In the following, in order to maximize the typicality of the instances of C, we minimize the instances of $\neg\Box\neg C$. Notice that this is different from maximising the instances of $\mathbf{T}(C)$. We have adopted this solution since it allows to maximise the set of typical instances of C without affecting

the extension of C (whereas maximising the extension of $\mathbf{T}(C)$ would imply maximising also the extension of C).

We define the set $\mathcal{M}^{\Box^-}_{\mathcal{L}_T}$ of negated boxed formulas holding in a model, relative to the concepts in $\mathcal{L}_T$. Given a model $\mathcal{M} = \langle \Delta, <, I \rangle$, let $\mathcal{M}^{\Box^-}_{\mathcal{L}_T} = \{(a, \neg\Box\neg C) \mid a \in (\neg\Box\neg C)^I, \text{ with } a \in \Delta, C \in \mathcal{L}_T\}$.

Let KB be a knowledge base and let $\mathcal{L}_T$ be a set of concepts occurring in KB.

Definition 3 (Preferred and minimal models). *Given a model* $\mathcal{M} = \langle \Delta_\mathcal{M}, <_\mathcal{M}, I_\mathcal{M} \rangle$ *of* KB *and a model* $\mathcal{N} = \langle \Delta_\mathcal{N}, <_\mathcal{N}, I_\mathcal{N} \rangle$ *of* KB*, we say that* $\mathcal{M}$ *is preferred to* $\mathcal{N}$ *with respect to* $\mathcal{L}_T$*, and we write* $\mathcal{M} <_{\mathcal{L}_T} \mathcal{N}$*, if the following conditions hold:* $\Delta_\mathcal{M} = \Delta_\mathcal{N}$ *and* $\mathcal{M}^{\Box^-}_{\mathcal{L}_T} \subset \mathcal{N}^{\Box^-}_{\mathcal{L}_T}$*. A model* $\mathcal{M}$ *is a* minimal model *for* KB *(with respect to* $\mathcal{L}_T$*) if it is a model of* KB *and there is no a model* $\mathcal{M}'$ *of* KB *such that* $\mathcal{M}' <_{\mathcal{L}_T} \mathcal{M}$*.*

Definition 4 (Minimal Entailment in $\mathcal{ALC}+\mathbf{T}_{min}$). *A query* F *(see section below) is minimally entailed from a knowledge base* KB *with respect to* $\mathcal{L}_T$ *if it holds in all models of* KB *minimal with respect to* $\mathcal{L}_T$*. We write* KB $\models^{\mathcal{L}_T}_{min} F$*.*

While the original $\mathcal{ALC}+\mathbf{T}$ is *monotonic* (see [7]), $\mathcal{ALC}+\mathbf{T}_{min}$ is *nonmonotonic*.

Consider the following example: KB= $\{\mathbf{T}(S) \sqsubseteq \neg P, S(a), W(a)\}$ and $\mathcal{L}_T = \{S, W\}$. We have KB $\models^{\mathcal{L}_T}_{min} \neg P(a)$. Indeed, there is a unique minimal model of KB on the domain $\Delta = \{a\}$, in which a is an instance of $\mathbf{T}(S)$ (as well as an instance of $\mathbf{T}(W)$), and hence $\neg P$ holds in a. Observe that $\neg P(a)$ is obtained in spite of the presence of the irrelevant property $W(a)$.

Consider the knowledge base KB' obtained by adding to KB the formula $\mathbf{T}(S \sqcap W) \sqsubseteq P$ and to $\mathcal{L}_T$ concept $S \sqcap W$. From KB', $\neg P(a)$ is not derivable any more. Instead, we have that KB' $\models^{\mathcal{L}_T}_{min} P(a)$. KB' has a unique minimal model on the domain $\Delta = \{a, b\}$, in which a is an instance of $\mathbf{T}(S \sqcap W)$ and $\mathbf{T}(W)$, but is not an instance of $\mathbf{T}(S)$ (as there is a b, such that $b < a$ and S holds at b). This example shows that, in case of conflict (here, a cannot be both a typical instance of S and $S \sqcap W$), typicality in the more specific concept is preferred.

In general, a knowledge base KB may have no minimal model or more than one minimal model, with respect to a given $\mathcal{L}_T$. The following properties hold:

Proposition 1. *(i) If* KB *has a model, then* KB *has a minimal model with respect to any* $\mathcal{L}_T$*. (ii) Let us replace all inclusions of the form* $\mathbf{T}(C) \sqsubseteq C'$ *in* KB *with* $C \sqsubseteq C'$*, and call* KB' *the resulting knowledge base. If* KB $\models^{\mathcal{L}_T}_{min} F$ *then* KB' $\models_{\mathcal{ALC}+\mathbf{T}} F$*, where* F *is a query.*

4 Reasoning

In this section we present a tableau calculus for deciding whether a formula (*query*) F is minimally entailed from a knowledge base (TBox,ABox). We introduce a labelled tableau calculus called $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$, which extends the calculus $T^{\mathcal{ALC}+\mathbf{T}}$ presented in [7], and allows to reason about minimal models.

$\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ performs a two-phase computation in order to check whether a query F is minimally entailed from the initial KB. In particular, the procedure tries to build an open branch representing a minimal model satisfying KB $\cup \{\neg F\}$. A query F is either a formula of the form $C(a)$ or a subsumption relation $C \sqsubseteq D$ such that, for all $\mathbf{T}(C')$ occurring in F, $C' \in \mathcal{L}_T$. In the first phase, a tableau calculus, called $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, simply verifies whether KB $\cup\{\neg F\}$ is satisfiable in an $\mathcal{ALC}+\mathbf{T}$ model, building candidate models. In case F has the form $C \sqsubseteq D$, then $\neg F$ corresponds to $(C \sqcap \neg D)(x)$, where x does not occur in KB. In the second phase another tableau calculus, called $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$, checks whether the candidate models found in the first phase are *minimal* models of KB, i.e. for each open branch of the first phase, $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ tries to build a "smaller" model of KB, i.e. a model whose individuals satisfy less formulas $\neg\Box\neg C$ than the corresponding candidate model. The whole procedure $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ is described at the end of this section (Definition 8).

$\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ is based on the notion of a *constraint system*. We consider a set of *variables* drawn from a denumerable set $\mathcal{V}$. $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ makes use of labels, which are denoted with $x, y, z, \ldots$. Labels represent *objects*. An object is either a variable or an individual of the ABox, that is to say an element of $\mathcal{O} \cup \mathcal{V}$. A *constraint* is a syntactic entity of the form $x \xrightarrow{R} y$ or $x : C$, where R is a role and C is either an extended concept or has the form $\Box\neg D$ or $\neg\Box\neg D$, where D is a concept. A constraint of the form $x \xrightarrow{R} y$ says that the object denoted by label x is related to the object denoted by y by means of role R; a constraint $x : C$ says that the object denoted by x is an instance of the concept C.

Let us now separately analyze the two components of the calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$, starting with $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$.

A tableau of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ is a tree whose nodes are pairs $\langle S \mid U\rangle$, where S contains constraints (or *labelled* formulas) of the form $x : C$ or $x \xrightarrow{R} y$, whereas U contains formulas of the form $C \sqsubseteq D^L$, representing subsumption relations $C \sqsubseteq D$ of the TBox. L is a list of labels. As we will discuss later, this list is used in order to ensure the termination of the tableau calculus.

A node $\langle S \mid U\rangle$ is also called a *constraint system*. A branch is a sequence of nodes $\langle S_1 \mid U_1\rangle, \langle S_2 \mid U_2\rangle, \ldots, \langle S_n \mid U_n\rangle \ldots$, where each node $\langle S_i \mid U_i\rangle$ is obtained from its immediate predecessor $\langle S_{i-1} \mid U_{i-1}\rangle$ by applying a rule of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, having $\langle S_{i-1} \mid U_{i-1}\rangle$ as the premise and $\langle S_i \mid U_i\rangle$ as one of its conclusions. A branch is closed if one of its nodes is an instance of (Clash), otherwise it is open. A tableau is closed if all its branches are closed.

In order to check the satisfiability of a KB, we build the corresponding constraint system $\langle S \mid U\rangle$, and we check its satisfiability.

Definition 5 (Corresponding constraint system). *Given a knowledge base* KB=(*TBox,ABox*)*, we define its* corresponding constraint system $\langle S \mid U\rangle$ *as follows:* $S = \{a : C \mid C(a) \in ABox\} \cup \{a \xrightarrow{R} b \mid aRb \in ABox\}$ *and* $U = \{C \sqsubseteq D^{\emptyset} \mid C \sqsubseteq D \in TBox\}$.

Definition 6 (Model satisfying a constraint system). *Let $\mathcal{M}$ be a model as defined in Definition 1. We define a function α which assigns to each variable of*

$\mathcal{V}$ an element of Δ, and assigns every individual $a \in \mathcal{O}$ to $a^I \in \Delta$. $\mathcal{M}$ satisfies $x : C$ under α if $\alpha(x) \in C^I$ and $x \xrightarrow{R} y$ under α if $(\alpha(x), \alpha(y)) \in R^I$. A constraint system $\langle S \mid U \rangle$ is satisfiable if there is a model $\mathcal{M}$ and a function α such that $\mathcal{M}$ satisfies every constraint in S under α and that, for all $C \sqsubseteq D^L \in U$ and for all x occurring in S, we have that if $\alpha(x) \in C^I$ then $\alpha(x) \in D^I$.

Proposition 2. *Given a knowledge base, it is satisfiable if and only if its corresponding constraint system is satisfiable.*

To verify the satisfiability of KB $\cup\{\neg F\}$, we use $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ to check the satisfiability of the constraint system $\langle S \mid U \rangle$ obtained by adding the constraint corresponding to $\neg F$ to S', where $\langle S' \mid U \rangle$ is the corresponding constraint system of KB. To this purpose, the rules of the calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ are applied until either a contradiction is generated (Clash) or a model satisfying $\langle S \mid U \rangle$ can be obtained from the resulting constraint system.

As in the calculus proposed in [7], given a node $\langle S \mid U \rangle$, for each subsumption $C \sqsubseteq D^L \in U$ and for each label x that appears in the tableau, we add to S the constraint $x : \neg C \sqcup D$ (*unfolding*). As mentioned above, each formula $C \sqsubseteq D$ is equipped with a list L of labels in which it has been unfolded in the current branch. This is needed to avoid multiple unfolding of the same subsumption by using the same label, generating infinite branches.

Before introducing the rules of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ we need some more definitions. First, as in [4], we define an ordering relation $\prec$ to keep track of the temporal ordering of insertion of labels in the tableau, that is to say if y is introduced in the tableau, then $x \prec y$ for all labels x that are already in the tableau.

Given a tableau node $\langle S \mid U \rangle$ and an object x, we define $\sigma(\langle S \mid U \rangle, x) = \{C \mid x : C \in S\}$. Furthermore, we say that two labels x and y are *S-equivalent*, written $x \equiv_S y$, if they label the same set of concepts, i.e. $\sigma(\langle S \mid U \rangle, x) = \sigma(\langle S \mid U \rangle, y)$. Intuitively, S-equivalent labels represent the same element in the model built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$. Last, we define $S^M_{x \to y} = \{y : \neg C, y : \Box\neg C \mid x : \Box\neg C \in S\}$.

The rules of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ are presented in Figure 1. Rules $(\exists^+)$ and $(\Box^-)$ are called *dynamic* since they introduce a new variable in their conclusions. The other rules are called *static*. The side condition on $(\exists^+)$ is introduced in order to ensure a terminating proof search, by implementing the standard *blocking* technique described below. The (*cut*) rule ensures that, given any concept $C \in \mathcal{L}_T$, an open branch built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ contains either $x : \Box\neg C$ or $x : \neg\Box\neg C$ for each label x: this is needed in order to allow $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ to check the minimality of the model corresponding to the open branch, as we will discuss later.

The rules of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ are applied with the following *standard strategy*: 1. apply a rule to a label x only if no rule is applicable to a label y such that $y \prec x$; 2. apply dynamic rules ($(\Box^-)$ first) only if no static rule is applicable. This strategy ensures that the labels are considered one at a time according to the ordering $\prec$. The calculus so obtained is sound and complete with respect to the semantics in Definition 6.

Theorem 1 (Soundness and Completeness of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$). *Given a constraint system $\langle S \mid U \rangle$, it is unsatisfiable iff it has a closed tableau in $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$.*

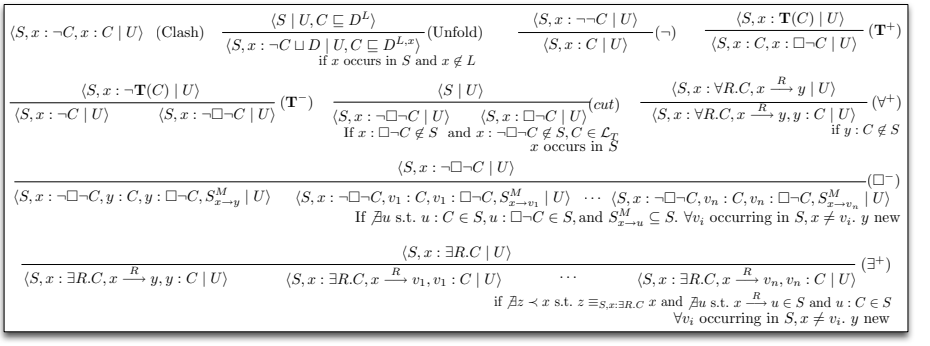


Fig. 1. The calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$. To save space, we omit the standard rules for $\sqcup$ and $\sqcap$, as well as the rules $(\forall^-)$ and $(\exists^-)$, dual to $(\exists^+)$ and $(\forall^+)$, respectively.

To ensure termination, we adopt the standard loop-checking machinery known as *blocking*: the side condition of the $(\exists^+)$ rule says that this rule can be applied to a node $\langle S, x : \exists R.C \mid U\rangle$ only if there is no z occurring in S such that $z \prec x$ and $z \equiv_{S,x:\exists R.C} x$. In other words, if there is an "older" label z which is equivalent to x wrt $S, x : \exists R.C$, then $(\exists^+)$ is not applicable, since the condition and the strategy imply that the $(\exists^+)$ rule has already been applied to z. In this case, we say that x is *blocked* by z. By virtue of the properties of $\Box$, no other additional machinery is required to ensure termination. Indeed, we can show that the interplay between rules $(\mathbf{T}^-)$ and $(\Box^-)$ does not generate branches containing infinitely-many labels. Intuitively, the application of $(\Box^-)$ to $x : \neg\Box\neg C$ adds $y : \Box\neg C$ to the conclusion, so that $(\mathbf{T}^-)$ can no longer consistently introduce $y : \neg\Box\neg C$. This is due to the properties of $\Box$ (no infinite descending chains of $<$ are allowed). The (*cut*) rule does not affect termination, since it is applied only to the finitely many formulas belonging to $\mathcal{L}_T$.

Theorem 2 (Termination of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$). *Let $\langle S \mid U\rangle$ be a constraint system, then any tableau generated by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ is finite.*

Since $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ is sound and complete (Theorem 1), and since a KB is satisfiable iff its corresponding constraint system is satisfiable (Proposition 2), from Theorem 2 above it follows that checking whether a given KB (TBox,ABox) is satisfiable is a decidable problem. It is possible to prove that, with the calculus above, the satisfiability of a KB can be decided in nondeterministic exponential time in the size of the KB.

Let us now introduce the calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ which, for each open branch $\mathbf{B}$ built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, verifies if it is a minimal model of the KB.

Definition 7. *Given an open branch $\mathbf{B}$ of a tableau built from $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, we define: (i) $\mathcal{D}(\mathbf{B})$ as the set of objects occurring on $\mathbf{B}$; (ii) $\mathbf{B}^{\Box^-} = \{x : \neg\Box\neg C \mid x : \neg\Box\neg C$ occurs in $\mathbf{B}\}$.*

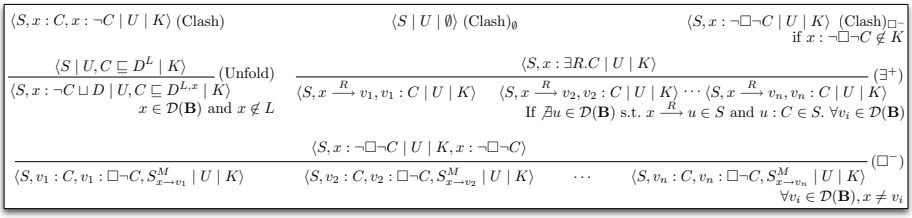


Fig. 2. The calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$

A tableau of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ is a tree whose nodes are triples of the form $\langle S \mid U \mid K\rangle$, where $\langle S \mid U\rangle$ is a constraint system, whereas K contains formulas of the form $x : \neg\Box\neg C$, with $C \in \mathcal{L}_T$.

The basic idea of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ is as follows. Given an open branch $\mathbf{B}$ built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ and corresponding to a model $\mathcal{M}^{\mathbf{B}}$ of KB $\cup\{\neg F\}$, $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ checks whether $\mathcal{M}^{\mathbf{B}}$ is a minimal model of KB by trying to build a model of KB which is preferred to $\mathcal{M}^{\mathbf{B}}$. Checking (un)satisfiability of $\langle S \mid U \mid \mathbf{B}^{\Box^-}\rangle$, where $\langle S \mid U\rangle$ is the corresponding constraint system of the initial KB, allows to verify whether the candidate model $\mathcal{M}^{\mathbf{B}}$ is minimal. More in detail, $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ tries to build an open branch containing all the objects appearing on $\mathbf{B}$, i.e. those in $\mathcal{D}(\mathbf{B})$. To this aim, the dynamic rules use labels in $\mathcal{D}(\mathbf{B})$ instead of introducing new ones in their conclusions. The additional set K of a tableau node, initialized with $\mathbf{B}^{\Box^-}$, is used in order to ensure that any branch **B'** built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ is preferred to $\mathbf{B}$, that is **B'** only contains negated boxed formulas occurring in $\mathbf{B}$ and there exists at least a $x : \neg\Box\neg C$ that occurs in $\mathbf{B}$ and *does not occur* in **B'**. The rules of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ are shown in Figure 2. $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ also contains analogues of the following rules from $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ in Figure 1: $(\neg)$, $(\mathbf{T}^+)$, $(\mathbf{T}^-)$, (cut), $(\forall^+)$, where the rules in $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ include the additional component K.

More in detail, the rule $(\exists^+)$ is applied to a constraint system containing a formula $x : \exists R.C$; it introduces $x \xrightarrow{R} y$ and $y : C$ where $y \in \mathcal{D}(\mathbf{B})$, instead of y being a new label. The choice of the label y introduces a branching in the tableau construction. The rule (Unfold) is applied in the same way as in $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ to *all the labels of* $\mathcal{D}(\mathbf{B})$ (and not only to those appearing in the branch). The rule $(\Box^-)$ is applied to a node $\langle S, x : \neg\Box\neg C \mid U \mid K\rangle$, when $x : \neg\Box\neg C \in K$, i.e. when the formula $x : \neg\Box\neg C$ also belongs to the open branch $\mathbf{B}$. In this case, the rule introduces a branch on the choice of the individual $v_i \in \mathcal{D}(\mathbf{B})$ which is preferred to x and is such that C and $\Box\neg C$ hold in v_i. In case a tableau node has the form $\langle S, x : \neg\Box\neg C \mid U \mid K\rangle$, and $x : \neg\Box\neg C \notin K$, then $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ detects a clash, called $(\text{Clash})_{\Box^-}$: this corresponds to the situation in which $x : \neg\Box\neg C$ does not belong to $\mathbf{B}$, while $S, x : \neg\Box\neg C$ is satisfiable in a model $\mathcal{M}$ only if $\mathcal{M}$ contains $x : \neg\Box\neg C$, and hence only if $\mathcal{M}$ is *not* preferred to the model represented by $\mathbf{B}$.

The calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ also contains the clash condition $(\text{Clash})_{\emptyset}$. Since each application of $(\Box^-)$ removes the principal formula $x : \neg\Box\neg C$ from the set K, when K is empty all the negated boxed formulas occurring in $\mathbf{B}$ also belong

to the current branch. In this case, the model built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ satisfies the same set of negated boxed formulas (for all individuals) as $\mathbf{B}$ and, thus, it is not preferred to the one represented by $\mathbf{B}$.

Theorem 3 (Soundness and completeness of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$). *Given a* KB *and a query F, let $\langle S' \mid U\rangle$ be the corresponding constraint system of* KB*, and $\langle S \mid U\rangle$ the corresponding constraint system of* KB $\cup\{\neg F\}$*. An open branch* $\mathbf{B}$ *built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ for $\langle S \mid U\rangle$ is satisfiable by an injective mapping in a minimal model of* KB *iff the tableau in $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ for $\langle S' \mid U \mid \mathbf{B}^{\Box^-}\rangle$ is closed.*

$\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ always terminates. Indeed, only a finite number of labels can occur on the branch (only those in $\mathcal{D}(\mathbf{B})$ which is finite). Moreover, the side conditions on the application of the rules copying their principal formulas in their conclusion(s) prevent the uncontrolled application of the same rules.

It is possible to show that the problem of verifying that a branch $\mathbf{B}$ represents a minimal model for KB in $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ is in NP in the size of $\mathbf{B}$.

The overall procedure $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ is defined as follows:

Definition 8. *Let KB be a knowledge base whose corresponding constraint system is $\langle S \mid U\rangle$. Let F be a query and let S' be the set of constraints obtained by adding to S the constraint corresponding to $\neg F$. The calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ checks whether a query F can be minimally entailed from a* KB *by means of the following procedure:* (phase 1) *the calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ is applied to $\langle S' \mid U\rangle$; if, for each branch* $\mathbf{B}$ *built by $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$, either (i)* $\mathbf{B}$ *is closed or (ii)* (phase 2) *the tableau built by the calculus $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ for $\langle S \mid U \mid \mathbf{B}^{\Box^-}\rangle$ is open, then* KB $\models^{\mathcal{L}_T}_{min} F$*, otherwise* KB $\not\models^{\mathcal{L}_T}_{min} F$*.*

$\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$ is therefore a sound and complete decision procedure for verifying if a formula F can be minimally entailed from a KB. We can also prove that:

Theorem 4 (Complexity of $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{min}$). *The problem of deciding whether* KB $\models^{\mathcal{L}_T}_{min} F$ *is in* co-NExp$^{\text{NP}}$*.*

As an example, let KB contain $\{\mathbf{T}(C) \sqsubseteq \neg P, C(a), D(a)\}$. We show that KB $\models^{\mathcal{L}_T}_{min} \neg P(a)$ with $\mathcal{L}_T = \{C\}$. The tableau $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH1}$ is initialised with $\langle S \cup \{a : \neg\neg P\} \mid U\rangle$, where $S = \{a : C, a : D\}$ and $U = \{\mathbf{T}(C) \sqsubseteq \neg P^{\emptyset}\}$. We apply (Unfold), ($\sqcup^+$) and then the ($\mathbf{T}^-$) rule. Disregarding the nodes that contain a clash, the only left branch $\mathbf{B}$ contains (after the application of the ($\neg$) rule) as lowest node: $\langle S' \mid U'\rangle$, where $U' = \{\mathbf{T}(C) \sqsubseteq \neg P^{\{a\}}\}$ and $S' = \{a : C, a : D, a : P, a : \neg\Box\neg C\}$. $\mathbf{B}$ may be further expanded. By applying ($\Box^-$), (Unfold), ($\sqcup^+$) and then ($\mathbf{T}^-$) we can generate only one open extension of $\mathbf{B}$ and it contains: $S'' = \{a : C, a : D, a : P, a : \neg\Box\neg C, b : C, b : \Box\neg C, b : \neg P\}$. We now apply the procedure $\mathcal{TAB}^{\mathcal{ALC}+\mathbf{T}}_{PH2}$ to $\mathbf{B}$: the tableau is initialised with $\langle S \mid U \mid K\rangle$, where $K = \{a : \neg\Box\neg C\}$; its expansion contains a branch whose lowest node is $\langle S_1 \mid U' \mid K\rangle$, where $S_1 = \{a : C, a : D, a : \Box\neg C, a : \neg P\}$ and U' is as above. This node can be further expanded by applying (Unfold) wrt. $b \in \mathcal{D}(\mathbf{B})$ and ($\mathbf{T}^-$) obtaining three nodes $\langle S_{1,1} \mid U'' \mid K\rangle$, $\langle S_{1,2} \mid U'' \mid K\rangle$, $\langle S_{1,3} \mid U'' \mid K\rangle$ where

$S_{1,1} = S_1 \cup \{b : \neg C\}$, $S_{1,2} = S_1 \cup \{b : \neg\Box\neg C\}$, $S_{1,3} = S_1 \cup \{b : \neg P\}$ and $U'' = \{\mathbf{T}(C) \sqsubseteq \neg P^{\{a,b\}}\}$. The node $\langle S_{1,2} \mid U'' \mid K\rangle$ is closed by $(\text{Clash})_{\Box-}$, the two others may be expanded by (*cut*) on $b : (\neg)\Box\neg C$, obtaining finally the following open nodes: $\langle S_{1,1} \cup \{b : \Box\neg C\} \mid U'' \mid K\rangle$ and $\langle S_{1,3} \cup \{b : \Box\neg C\} \mid U'' \mid K\rangle$. Since the two nodes are open, the tableau $\mathcal{TAB}_{PH2}^{\mathcal{ALC}+\mathbf{T}}$ for $\mathbf{B}$ is open, whence $\mathbf{B}$ is closed. Thus the whole procedure $\mathcal{TAB}_{min}^{\mathcal{ALC}+\mathbf{T}}$ verifies that KB $\models_{min}^{\mathcal{L}_T} \neg P(a)$.

5 Discussion and Conclusion

We have proposed a nonmonotonic extension called $\mathcal{ALC} + \mathbf{T}_{min}$ of $\mathcal{ALC}$ for reasoning about prototypical properties in Description Logic framework. The extension is obtained by adding first a typicality operator, originally defined in [7], to $\mathcal{ALC}$. This extension, called $\mathcal{ALC} + \mathbf{T}$, provides a monotonic extension of $\mathcal{ALC}$ that enjoys a simple modal semantics. One advantage of the use of a typicality operator is that we can express prototypical properties directly in the form "the most typical instances of concept C have the property P" (corresponding to $\mathbf{T}(C) \sqsubseteq P$), as opposed to rather complicated encodings within other nonmonotonic formalisms. However, $\mathcal{ALC} + \mathbf{T}$ is not sufficient to perform defeasible reasoning. For this reason in the present work we develop a preferential semantics, called $\mathcal{ALC} + \mathbf{T}_{min}$. This nonmonotonic extension allows one to perform defeasible reasoning in particular in the context of inheritance with exceptions to some extent. We have then developed a procedure for deciding query-entailment in $\mathcal{ALC} + \mathbf{T}_{min}$. The procedure has the form of a two-phase tableau calculus for generating $\mathcal{ALC} + \mathbf{T}_{min}$ minimal models. The procedure is sound, complete, and terminating, whereby giving a decision procedure for deciding $\mathcal{ALC} + \mathbf{T}_{min}$ entailment in co-NEXP$^{\text{NP}}$.

We plan to extend this work in several directions. First of all, the tableau procedure we have described can be optimised in many ways. For instance, we guess that the calculus $\mathcal{TAB}_{PH1}^{\mathcal{ALC}+\mathbf{T}}$, dealing with the monotonic logic $\mathcal{ALC} + \mathbf{T}$, can be made more efficient by applying standard techniques such as caching. More precisely, we expect to obtain an EXP decision procedure. Furthermore, the (*cut*) rule could be applied in $\mathcal{TAB}_{PH2}^{\mathcal{ALC}+\mathbf{T}}$ by distinguishing if $x : \neg\Box\neg C$ belongs to K or not: in the second case, the branch where $x : \neg\Box\neg C$ is introduced is closed by $(\text{Clash})_{\Box-}$. Although the calculus $\mathcal{TAB}_{min}^{\mathcal{ALC}+\mathbf{T}}$ provides a decision procedure, we have still to determine the exact complexity of $\mathcal{ALC} + \mathbf{T}_{min}$ itself.

From the point of view of knowledge representation, a limit of our logic is the unability to handle inheritance of multiple properties in case of exceptions as in the example: $\mathbf{T}(\mathit{Student}) \sqsubseteq \neg\mathit{HasIncome}$, $\mathbf{T}(\mathit{Student}) \sqsubseteq \exists \mathit{Owns}.\mathit{LibraryCard}$, $\mathit{PhDStudent} \sqsubseteq \mathit{Student}$, $\mathbf{T}(\mathit{PhDStudent}) \sqsubseteq \mathit{HasIncome}$. Our semantics does not support the inference $\mathbf{T}(\mathit{PhDStudent}) \sqsubseteq \exists \mathit{Owns}.\mathit{LibraryCard}$, that is PhDStudents typically own a library card, as we might want to conclude (since having an income has nothing to do with owning a library card). The reason why our semantics fails to support this inference is that the first two inclusions are obviously equivalent to the single one $\mathbf{T}(\mathit{Student}) \sqsubseteq \neg\mathit{HasIncome} \sqcap \exists \mathit{Owns}.\mathit{LibraryCard}$ which is contradicted by $\mathbf{T}(\mathit{PhDStudent}) \sqsubseteq \mathit{HasIncome}$. To handle this type of

inferences we would need a tighter semantics where the truth of $\mathbf{T}(C) \sqsubseteq P$ is no longer a function of $\mathbf{T}(C)$ and P or a smarter (and less direct) encoding of the knowledge. Observe that the same problem arises for instance with circumscription, where we would need at least different abnormality predicates *for each pair* of concept-defeasible property. This problem is perhaps better addressed by probabilistic extensions of description logics such as [8].

KLM logics, which are at the base of our semantics, are related to probabilistic reasoning. In [8], the notion of conditional constraint allows typicality assertions to be expressed (with a specified interval of probability values). In order to perform defeasible reasoning, a notion of minimal entailment is introduced based on a *lexicographic preference* relation on probabilistic interpretations. We plan to compare in details this probabilistic approach to ours in further research.

Moreover, we intend to investigate the precise relation of $\mathcal{ALC} + \mathbf{T}_{min}$ with other formalisms for nonmonotonic reasoning, first of all with circumscription. To this regard, Moinard in [11] has shown that several kinds of preferential entailment (in propositional logic) can be translated into generalised forms of circumscription by extending the vocabulary, as a matter of fact, to an exponentially larger vocabulary. It might be worth investigating if a similar encoding works in our case. More concretely, we may wonder if there is a direct translation of knowledge bases from $\mathcal{ALC}+\mathbf{T}_{min}$ to circumscription and viceversa. As a starting point, concerning the direction from $\mathcal{ALC}+\mathbf{T}_{min}$ to circumscription, we guess that the translation should map every subsumption relation $\mathbf{T}(C_i) \sqsubseteq Q_i$ of a KB into a subsumption relation of the kind $C_i \sqsubseteq Q_i \sqcup Abn_{C_i}$, where, for each concept C_i, Abn_{C_i} is a distinct abnormality concept name to minimize. Then we may ask whether there exists a circumscription pattern [3] CP where (i) all concept names different from Abn_{C_i} and all roles vary and (ii) concept names Abn_{C_i} are minimised according to a suitable partial order (to be discovered), such that the queries entailed by the KB in $\mathcal{ALC} + \mathbf{T}_{min}$ coincide with the queries entailed by the translated KB under circumscription with respect to pattern CP. We shall deal with this and related questions in future research.

References

1. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. J. Autom. Reasoning 14(1), 149–180 (1995)
2. Baader, F., Hollunder, B.: Priorities on defaults with prerequisites, and their application in treating specificity in terminological default logic. J. of Automated Reasoning (JAR) 15(1), 41–68 (1995)
3. Bonatti, P.A., Lutz, C., Wolter, F.: Description logics with circumscription. In: Proc. of KR, pp. 400–410 (2006)
4. Buchheit, M., Donini, F.M., Schaerf, A.: Decidable reasoning in terminological knowledge representation systems. J. Artif. Int. Research (JAIR) 1, 109–138 (1993)
5. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Trans. Comput. Log. 3(2), 177–225 (2002)
6. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. In: KR 2004, pp. 141–151 (2004)

7. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Preferential Description Logics. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 257–272. Springer, Heidelberg (2007)
8. Giugno, R., Lukasiewicz, T.: P-$\mathcal{SHOQ}$(D): A Probabilistic Extension of $\mathcal{SHOQ}$(D) for Probabilistic Ontologies in the Semantic Web. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 86–97. Springer, Heidelberg (2002)
9. Ke, P., Sattler, U.: Next Steps for Description Logics of Minimal Knowledge and Negation as Failure. In: DL 2008 (2008)
10. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
11. Moinard, Y.: General Preferential Entailments as Circumscriptions. In: Benferhat, S., Besnard, P. (eds.) ECSQARU 2001. LNCS (LNAI), vol. 2143, pp. 532–543. Springer, Heidelberg (2001)
12. Straccia, U.: Default inheritance reasoning in hybrid kl-one-style logics. In: Proc. of IJCAI, pp. 676–681 (1993)

Counting Complexity of Minimal Cardinality and Minimal Weight Abduction*

Miki Hermann[1] and Reinhard Pichler[2]

[1] LIX (CNRS, UMR 7161), École Polytechnique, 91128 Palaiseau, France
hermann@lix.polytechnique.fr
[2] Institut für Informationssysteme, Technische Universität Wien, A-1040 Wien, Austria
pichler@dbai.tuwien.ac.at

Abstract. Abduction is an important method of non-monotonic reasoning with many applications in artificial intelligence and related topics. In this paper, we concentrate on propositional abduction, where the background knowledge is given by a propositional formula. We have recently started to study the counting complexity of propositional abduction. However, several important cases have been left open, namely, the cases when we restrict ourselves to solutions with minimal cardinality or with minimal weight. These cases – possibly combined with priorities – are now settled in this paper. We thus arrive at a complete picture of the counting complexity of propositional abduction.

1 Introduction

Abduction is a method of non-monotonic reasoning which has taken a fundamental importance in artificial intelligence and related topics. It aims at giving explanations for observed symptoms and is, therefore, widely used in diagnosis – notably in the medical domain (see [17]). Other important applications of abduction can be found in planning, database updates, data-mining and many more areas (see e.g. [11,12,16]).

Logic-based abduction is formally described as follows. Given a logical theory T, a set M of manifestations, and a set H of hypotheses, find a solution $\mathcal{S}$, i.e., a set $S \subseteq H$ such that $T \cup S$ is consistent and logically entails M. In this paper, we consider *propositional abduction problems* (PAPs, for short), where the theory T is represented by a propositional formula over a Boolean algebra $\mathbb{B} = (\{0,1\}; \vee, \wedge, \neg, \rightarrow, \equiv)$ and the sets H and M consist of variables from some set V. A *diagnosis problem* can be represented by a PAP $\mathcal{P} = \langle V, H, M, T \rangle$ as follows: The theory T is the system description. The hypotheses $H \subseteq V$ describe the possibly faulty system components. The manifestations $M \subseteq V$ are the observed symptoms, describing the malfunction of the system. The solutions $\mathcal{S}$ of $\mathcal{P}$ are the possible explanations of the malfunction.

Example 1. Consider the following football knowledge base.

$$\begin{aligned} T = \{\ & weak\_defense \wedge weak\_attack \rightarrow match\_lost, \\ & match\_lost \rightarrow manager\_sad \wedge press\_angry \\ & star\_injured \rightarrow manager\_sad \wedge press\_sad\ \} \end{aligned}$$

* This work was partially supported by the Austrian Science Fund (FWF), project P20704-N18.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 206–218, 2008.

Moreover, let the set of observed manifestations and the set of hypotheses be

$$M = \{\, manager\_sad \,\}$$
$$H = \{\, star\_injured, weak\_defense, weak\_attack \,\}$$

This PAP has the following five abductive explanations (= "solutions").

$$\mathcal{S}_1 = \{\, star\_injured \,\}$$
$$\mathcal{S}_2 = \{\, weak\_defense, weak\_attack \,\}$$
$$\mathcal{S}_3 = \{\, weak\_attack, star\_injured \,\}$$
$$\mathcal{S}_4 = \{\, weak\_defense, star\_injured \,\}$$
$$\mathcal{S}_5 = \{\, weak\_defense, weak\_attack, star\_injured \,\}$$

Obviously, in the above example, not all solutions are equally intuitive. Indeed, for many applications, one is not interested in *all* solutions of a given PAP $\mathcal{P}$ but only in *all acceptable* solutions of $\mathcal{P}$. *Acceptable* in this context means *minimal* with respect to some preorder $\preceq$ on the powerset 2^H. Two natural preorders are subset-minimality and cardinality-minimality, where the preorder is $\subseteq$ and $\leq$, respectively. In Example 1, both $\mathcal{S}_1$ and $\mathcal{S}_2$ are subset-minimal but only $\mathcal{S}_1$ is cardinality-minimal. If we have a weight function on the hypotheses then we may define the acceptable solutions as the weight-minimal ones. This preorder (i.e., smaller or equal weight) is denoted as $\sqsubseteq$.

All three criteria $\subseteq$, $\leq$, and $\sqsubseteq$ can be further refined by a hierarchical organization of our hypotheses according to some *priorities* (cf. [5]). In this context, priorities may be used to represent a qualitative version of probability. The resulting preorder is denoted by $\subseteq_P$, $\leq_P$, and $\sqsubseteq_P$. For instance, suppose that for some reason we know that (for a specific team) *star_injured* is much less likely to occur than *weak_defense* and *weak_attack*. This judgment can be formalized by assigning lower priority to the former. Then $\mathcal{S}_2$ is the only minimal solution with respect to the preorders $\subseteq_P$ and $\leq_P$. Actually, in this simple example, $\mathcal{S}_2$ is also the only $\sqsubseteq_P$-minimal solution independently of the concrete weight function. Finally, if indeed all solutions are acceptable, then the corresponding preorder is the syntactic equality $=$.

The usually observed algorithmic problem in logic-based abduction is the existence problem, i.e. deciding whether at least one solution $\mathcal{S}$ exists for a given abduction problem $\mathcal{P}$. Another well-studied decision problem is the so-called relevance problem, i.e. Given a PAP $\mathcal{P}$ and a hypothesis $h \in H$, is h part of at least one acceptable solution? However, this approach is not always satisfactory. Especially in database applications, in diagnosis, and in data-mining there exist situations where we need to know *all* acceptable solutions of the abduction problem or at least an important part of them. Consequently, the enumeration problem (i.e., the computation of all acceptable solutions) has received much interest (see e.g. [3, 4]). Another natural question is concerned with the total number of solutions to the considered problem. The latter problem refers to the *counting complexity* of abduction. Clearly, the counting complexity provides a lower bound for the complexity of the enumeration problem. Moreover, counting the number of abductive explanations can be useful for probabilistic abduction problems (see e.g. [18]). Indeed, in order to compute the probability of failure of a given component in a diagnosis problem (under the assumption that all preferred explanations are equiprobable), we need to count the number of preferred explanations as well as the number of preferred explanations that contain a given hypothesis.

Table 1. Counting complexity of propositional abduction

#-Abduction	$=$	$\subseteq$	$\subseteq_P$	$\leq$	$\leq_P$	$\sqsubseteq, \sqsubseteq_P$
General case	$\#\cdot\mathrm{coNP}$	$\#\cdot\mathrm{coNP}$	$\#\cdot\Pi_2\mathrm{P}$	$\#\cdot\mathrm{Opt}_2\mathrm{P}[\log n]$	$\#\cdot\mathrm{Opt}_2\mathrm{P}$	$\#\cdot\mathrm{Opt}_2\mathrm{P}$
Horn	$\#\mathrm{P}$	$\#\mathrm{P}$	$\#\cdot\mathrm{coNP}$	$\#\cdot\mathrm{OptP}[\log n]$	$\#\cdot\mathrm{OptP}$	$\#\cdot\mathrm{OptP}$
definite Horn	$\#\mathrm{P}$	$\#\mathrm{P}$	$\#\mathrm{P}$	$\#\cdot\mathrm{OptP}[\log n]$	$\#\cdot\mathrm{OptP}$	$\#\cdot\mathrm{OptP}$
dual Horn	$\#\mathrm{P}$	$\#\mathrm{P}$	$\#\mathrm{P}$	$\#\cdot\mathrm{OptP}[\log n]$	$\#\cdot\mathrm{OptP}$	$\#\cdot\mathrm{OptP}$
bijunctive	$\#\mathrm{P}$	$\#\mathrm{P}$	$\#\cdot\mathrm{coNP}$	$\#\cdot\mathrm{OptP}[\log n]$	$\#\cdot\mathrm{OptP}$	$\#\cdot\mathrm{OptP}$

The study of counting complexity has been initiated by Valiant [19, 20] and is now a well-established part of the complexity theory, where the best known class is $\#\mathrm{P}$. Many counting variants of decision problems have been proved $\#\mathrm{P}$-complete. Higher counting complexity classes do exist, but they are not commonly known. A counting equivalent of the polynomial hierarchy was defined by Hemaspaandra and Vollmer [8], whereas generic complete problems for these counting hierarchy classes were presented in [1]. We enlarged in [10] the approach of Hemaspaandra and Vollmer to classes of optimization problem, obtaining this way a new hierarchy of classes $\#\cdot\mathrm{Opt}_k\mathrm{P}[\log n]$ and $\#\cdot\mathrm{Opt}_k\mathrm{P}$ for arbitrary $k \in \mathbb{N}$. These classes are sandwiched between the previously known counting classes $\#\cdot\Pi_k\mathrm{P}$, i.e., for each $k \in \mathbb{N}$ we have

$$\#\cdot\Pi_k\mathrm{P} \subseteq \#\cdot\mathrm{Opt}_{k+1}\mathrm{P}[\log n] \subseteq \#\cdot\mathrm{Opt}_{k+1}\mathrm{P} \subseteq \#\cdot\Pi_{k+1}\mathrm{P}.$$

It was shown in [10] that these inclusions are proper unless the polynomial hierarchy collapses to the k-th level. The most important special case is $k = 1$, where we write $\#\cdot\mathrm{OptP}[\log n]$ and $\#\cdot\mathrm{OptP}$ as a short-hand for $\#\cdot\mathrm{Opt}_1\mathrm{P}[\log n]$ and $\#\cdot\mathrm{Opt}_1\mathrm{P}$. On the first two levels, we thus have the inclusions $\#\mathrm{P} \subseteq \#\cdot\mathrm{OptP}[\log n] \subseteq \#\cdot\mathrm{OptP} \subseteq \#\cdot\mathrm{coNP} \subseteq \#\cdot\mathrm{Opt}_2\mathrm{P}[\log n] \subseteq \#\cdot\mathrm{Opt}_2\mathrm{P} \subseteq \#\cdot\Pi_2\mathrm{P}$. It will turn out that these new counting complexity classes are precisely the ones needed to pinpoint the exact counting complexity of the open cases in propositional abduction.

Results. We considered in [9] propositional abduction counting problems with the three preorders $=$, $\subseteq$, and $\subseteq_P$. Together with the general case where T can be an arbitrary propositional formula, we also considered the special cases where T is Horn, definite Horn, dual Horn, and bijunctive. These are the most frequently studied subcases of propositional formulas. Our results from [9] are summarized in the first three columns of Table 1. In this paper we continue the investigation on counting complexity of propositional abduction, focusing on the preorders $\leq$, $\sqsubseteq$, $\leq_P$, and $\sqsubseteq_P$. Note that these are practically highly relevant cases for the following reasons: If the failure of any component in a system is independent of the failure of the other components and all components have equal failure probability, then explanations with minimum cardinality are the ones with highest probability. If we have numeric values available for the repair cost or for the robustness of each component (e.g., based on data such as the empirically collected mean time to failure and component age), then weight-minimal abduction seeks for the cheapest repair respectively for the most likely explanation. If in addition different sets of components can be ranked according to some criterion that is not well

suited for numeric values (like, e.g., a qualitative rather than a quantitative robustness measure of components, the accessibility of components, or how critical the failure of a certain component would be), then this ranking can be expressed by priorities on the hypotheses, for both the cardinality and weight minimal case. Our results obtained in this work are summarized in the last three columns of Table 1. In total, we have thus achieved a complete picture of the counting complexity of propositional abduction.

2 Preliminaries

2.1 Propositional Abduction

A *propositional abduction problem* (PAP) $\mathcal{P}$ consists of a tuple $\langle V, H, M, T\rangle$, where V is a finite set of *variables*, $H \subseteq V$ is the set of *hypotheses*, $M \subseteq V$ is the set of *manifestations*, and T is a consistent *theory* in the form of a propositional formula. A set $\mathcal{S} \subseteq H$ is a *solution* (also called *explanation*) to $\mathcal{P}$ if $T \cup \mathcal{S}$ is consistent and $T \cup \mathcal{S} \models M$ holds. *Priorities* $P = \langle H_1, \ldots, H_K\rangle$ are a stratification of the hypotheses $H = H_1 \cup \cdots \cup H_K$ into a fixed number of disjoint sets. The *minimal cardinality with priorities* relation $A \leq_P B$ holds if $A = B$ or there exists an $i \in \{1, \ldots, K\}$ such that $A \cap H_j = B \cap H_j$ for all $j < i$ and $|A \cap H_i| < |B \cap H_i|$. The *minimal weight with priorities* relation $A \sqsubseteq_P B$ holds if $A = B$ or there exists an $i \in \{1, \ldots, K\}$ such that $A \cap H_j = B \cap H_j$ for all $j < i$ and $\sum_{a \in A \cap H_i} w(a) < \sum_{b \in B \cap H_i} w(b)$, where $w\colon H \to \mathbb{N}$ is the weight function on the hypotheses H.

We study the following family of counting problems, which are parameterized by a preorder $\preceq$ on 2^H.

Problem: #-$\preceq$-ABDUCTION
Input: A propositional abduction problem $\mathcal{P} = \langle V, H, M, T\rangle$.
Output: Number of $\preceq$-minimal solutions (explanations) of $\mathcal{P}$.

We considered the abduction counting problems with the preorders of equality $=$, subset minimality $\subseteq$, and subset minimality with priorities $\subseteq_P$ in [9]. In this paper we consider the preorders of minimal cardinality $\leq$, minimal weight $\sqsubseteq$, as well as their versions with priorities $\leq_P$ and $\sqsubseteq_P$, respectively. It is clear that an upper bound for a minimal weight decision or counting abduction problem subsumes that for the corresponding abduction problem for minimal cardinality. Similarly, a lower bound for a minimal cardinality abduction problem subsumes that for minimal weight abduction. In both cases, setting the weight of each hypothesis $x \in H$ to $w(x) = 1$ corresponds to the cardinality version. Throughout this paper, we follow the formalism of Eiter and Gottlob [2], allowing only positive literals in the solutions.

Together with the general case where T can be an arbitrary propositional formula, we consider the special cases where T is Horn, definite Horn, dual Horn, and bijunctive. A propositional clause C is said to be *Horn*, *definite Horn*, *dual Horn*, or *bijunctive* if it has at most one positive literal, exactly one positive literal, at most one negative literal, or at most two literals, respectively. A theory T is Horn, definite Horn, dual Horn, or bijunctive if it is a conjunction (or, equivalently, a set) of Horn, definite Horn, dual Horn, or bijunctive, clauses, respectively.

2.2 Counting Complexity

The study of *counting problems* was initiated by Valiant in [19, 20]. While decision problems ask if at least one solution of a given problem instance exists, counting problems ask for the number of different solutions. The most intensively studied counting complexity class is $\#\mathrm{P}$, which denotes the functions that count the number of accepting paths of a non-deterministic polynomial-time Turing machine. In other words, $\#\mathrm{P}$ captures the counting problems corresponding to decision problems in NP. By allowing the non-deterministic polynomial-time Turing machine access to an oracle in NP, $\Sigma_2\mathrm{P}$, $\Sigma_3\mathrm{P}$, ..., we can define an infinite hierarchy of counting complexity classes.

Alternatively, a *counting problem* is presented using a *witness* function which for every input x returns a set of *witnesses* for x. A *witness* function is a function $w\colon \Sigma^* \to \mathcal{P}^{<\omega}(\Gamma^*)$, where Σ and Γ are two alphabets, and $\mathcal{P}^{<\omega}(\Gamma^*)$ is the collection of all finite subsets of Γ^*. Every such witness function gives rise to the following *counting problem*: given a string $x \in \Sigma^*$, find the cardinality $|w(x)|$ of the *witness* set $w(x)$. According to [8], if $\mathcal{C}$ is a complexity class of decision problems, we define $\#\cdot\mathcal{C}$ to be the class of all counting problems whose witness function w satisfies the following conditions.

1. There is a polynomial $p(n)$ such that for every $x \in \Sigma^*$ and every $y \in w(x)$ we have $|y| \leq p(|x|)$;
2. The problem "given x and y, is $y \in w(x)$?" is in $\mathcal{C}$.

It is easy to verify that $\#\mathrm{P} = \#\cdot\mathrm{P}$. The counting hierarchy is ordered by linear inclusion [8]. In particular, we have that $\#\mathrm{P} \subseteq \#\cdot\mathrm{coNP} \subseteq \#\cdot\Pi_2\mathrm{P} \subseteq \#\cdot\Pi_3\mathrm{P}$, etc

In [10] we introduced new counting complexity classes for counting *optimal* solutions. We followed the aforementioned approach, where the complexity class $\mathcal{C}$ was chosen among OptP and $\mathrm{OptP}[\log n]$, or, more generally, $\mathrm{Opt}_k\mathrm{P}$ and $\mathrm{Opt}_k\mathrm{P}[\log n]$ for arbitrary $k \in \mathbb{N}$, respectively. These classes were previously defined by Krentel [14, 15]. A large collection of completeness results for these classes is given in [7]. As Krentel observed, the classes $\mathrm{OptP}[\log n]$ and OptP, which are closely related to $\mathrm{FP}^{\mathrm{NP}[\log n]}$ and $\mathrm{FP}^{\mathrm{NP}}$, contain problems computing optimal solutions with a logarithmic and polynomial number of calls to an NP-oracle, respectively.

The application of the counting operator to the aforementioned optimization classes allowed us to define in [10] the counting complexity classes $\#\cdot\mathrm{OptP}$, $\#\cdot\mathrm{OptP}[\log n]$ and, more generally, $\#\cdot\mathrm{Opt}_k\mathrm{P}$, $\#\cdot\mathrm{Opt}_k\mathrm{P}[\log n]$ for each $k \in \mathbb{N}$. To formally introduce these classes, we need some supplementary notions.

A *non-deterministic transducer* M is a non-deterministic polynomial-time bounded Turing machine, which writes a binary number on the output at the end of every accepting path. If M is equipped with an oracle from the complexity class $\mathcal{C}$, then it is called a *non-deterministic transducer with $\mathcal{C}$-oracle*. A $\Sigma_k\mathrm{P}$-*transducer* M is a non-deterministic transducer with a $\Sigma_{k-1}\mathrm{P}$ oracle. We identify non-deterministic transducers without oracle and $\Sigma_1\mathrm{P}$-transducers. For $x \in \Sigma^*$, we write $\mathrm{opt}_M(x)$ to denote the *optimal* value, which can be either the *maximum* or the *minimum*, on any accepting path of the computation of M on x. If no accepting path exists then $\mathrm{opt}_M(x)$ is undefined.

We say that a counting problem $\#\cdot A\colon \Sigma^* \to \mathbb{N}$ is in the class $\#\cdot\mathrm{Opt}_k\mathrm{P}$ for some $k \in \mathbb{N}$, if there is a $\Sigma_k\mathrm{P}$-transducer M, such that $\#\cdot A(x)$ is the number of accepting paths of the computation of M on x yielding the optimum value $\mathrm{opt}_M(x)$. If

no accepting path exists then $\#\cdot A(x) = 0$. If the length of the binary number written by M is bounded by $O(\log |x|)$, then $\#\cdot A$ is in the class $\#\cdot\mathrm{Opt}_k\mathrm{P}[\log n]$. For $k = 1$, we write $\#\cdot\mathrm{OptP}[\log n]$ and $\#\cdot\mathrm{OptP}$ as a short-hand for $\#\cdot\mathrm{Opt}_1\mathrm{P}[\log n]$ and $\#\cdot\mathrm{Opt}_1\mathrm{P}$, respectively. It was shown in [10] that these new classes $\#\cdot\mathrm{Opt}_k\mathrm{P}[\log n]$ and $\#\cdot\mathrm{Opt}_k\mathrm{P}$ are robust, i.e., they do not collapse to already known counting complexity classes unless the polynomial hierarchy collapses as well. Finally, these new counting classes were shown to be sandwiched between the classes $\#\cdot\Pi_k\mathrm{P}$, i.e., we obtained the inclusions $\#\mathrm{P} \subseteq \#\cdot\mathrm{OptP}[\log n] \subseteq \#\cdot\mathrm{OptP} \subseteq \#\cdot\mathrm{coNP} \subseteq \#\cdot\mathrm{Opt}_2\mathrm{P}[\log n] \subseteq \#\cdot\mathrm{Opt}_2\mathrm{P} \subseteq \#\cdot\Pi_2\mathrm{P}$, etc.

The prototypical $\#\cdot\Pi_k\mathrm{P}$-complete problem for $k \in \mathbb{N}$ is $\#\Pi_k\mathrm{SAT}$ [1], defined as follows. Given a formula

$$\varphi(X) = \forall Y_1 \exists Y_2 \cdots Q_k Y_k \; \psi(X, Y_1, \ldots, Y_k)$$

where ψ is a Boolean formula and X, Y_1, …, Y_k are sets of propositional variables, count the number of truth assignments to the variables in X that satisfy φ. We obtain the prototypical $\#\cdot\mathrm{Opt}_{k+1}\mathrm{P}[\log n]$-complete problem #MIN-CARD-Π_kSAT and the prototypical $\#\cdot\mathrm{Opt}_{k+1}\mathrm{P}$-complete problem #MIN-WEIGHT-Π_kSAT [10] by asking for the number of cardinality-minimal and weight-minimal models of $\varphi(X)$. In the latter case, there exists a weight function $w\colon X \to \mathbb{N}$ assigning positive values to each variable $x \in X$. As usual, the counting problems #MIN-CARD-Π_0SAT and #MIN-WEIGHT-Π_0SAT are just denoted by #MIN-CARD-SAT and #MIN-WEIGHT-SAT, being respectively $\#\cdot\mathrm{OptP}[\log n]$- and $\#\cdot\mathrm{OptP}$-complete.

3 General Case

Theorem 1. *#-$\leq$-ABDUCTION is $\#\cdot\mathrm{Opt}_2\mathrm{P}[\log n]$-complete and #-$\sqsubseteq$-ABDUCTION is $\#\cdot\mathrm{Opt}_2\mathrm{P}$-complete.*

Proof. In order to prove the membership, we show that these problems can be solved by an appropriate $\Sigma_2\mathrm{P}$-transducer M, i.e., M works in non-deterministic polynomial time with access to an NP-oracle and, in case of #-$\leq$-ABDUCTION, the output of M is logarithmically bounded. We give a high-level description of M: It takes an arbitrary PAP $\mathcal{P} = \langle V, H, M, T\rangle$ as input and non-deterministically enumerates all subsets $\mathcal{S} \subseteq H$, such that every computation path of M corresponds to exactly one $\mathcal{S} \subseteq H$. By two calls to an NP-oracle, M checks on every path whether $T \cup \mathcal{S}$ is consistent (i.e., satisfiable) and if $T \cup \mathcal{S} \models M$ holds. If both oracle calls answer "yes", then $\mathcal{S}$ is a solution of $\mathcal{P}$ and the computation path is accepting. The output written by M on each path is the cardinality of the corresponding set $\mathcal{S}$ (resp. the sum of the weights of the elements in $\mathcal{S}$) for the #-$\leq$-ABDUCTION problem (resp. for the #-$\sqsubseteq$-ABDUCTION problem). Finally, we define the optimal value of M to be the minimum. Obviously, the accepting paths of M outputting the optimal value correspond one-to-one to the cardinality-minimal (resp. weight-minimal) solutions of the PAP $\mathcal{P}$.

The hardness of #-$\leq$-ABDUCTION (resp. of #-$\sqsubseteq$-ABDUCTION) is shown by reduction from #MIN-CARD-Π_1SAT (resp. from #MIN-WEIGHT-Π_1SAT). Let an arbitrary instance of #MIN-CARD-Π_1SAT (resp. of #MIN-WEIGHT-Π_1SAT) be given by the

quantified Boolean formula $\varphi(X) = \forall Y \ \psi(X, Y)$ with $X = \{x_1, \ldots, x_k\}$ and $Y = \{y_1, \ldots, y_l\}$. In case of #MIN-WEIGHT-Π_1SAT, we additionally have a weight function w defined on the variables in X. Let $X' = \{x'_1, \ldots, x'_k\}$, $X'' = \{x''_1, \ldots, x''_k\}$, $Q = \{q_1, \ldots, q_k\}$, $R = \{r_1, \ldots, r_k\}$, and t be fresh variables. Then we define the PAP $\mathcal{P} = \langle V, H, M, T \rangle$ as follows.

$$V = X \cup X' \cup X'' \cup Y \cup Q \cup R \cup \{t\}, \quad H = X \cup X' \cup X'', \quad M = Q \cup R \cup \{t\}$$
$$T = \{\psi(X, Y) \to t\} \cup \{\neg x_i \vee \neg x'_i, x_i \to q_i, x'_i \to q_i \mid i = 1, \ldots, k\}$$
$$\cup \{\neg x'_i \vee \neg x''_i, x'_i \to r_i, x''_i \to r_i \mid i = 1, \ldots, k\}.$$

In case of #-$\sqsubseteq$-ABDUCTION, we leave the weights of the variables in X unchanged. For the remaining hypotheses, we set $w(x_i) = w(x'_i) = w(x''_i)$ for every $i \in \{1, \ldots, k\}$.

For each i, the clauses $\neg x_i \vee \neg x'_i$, $x_i \to q_i$, $x'_i \to q_i$ in T ensure that every solution $\mathcal{S}$ of $\mathcal{P}$ contains exactly one of $\{x_i, x'_i\}$. Similarly, the clauses $\neg x'_i \vee \neg x''_i$, $x'_i \to r_i$, $x''_i \to r_i$ ensure that every solution contains exactly one of $\{x'_i, x''_i\}$. The sets of variables X' and X'' both represent the complement $X \setminus A$, but X'' is there to get the cardinalities right, since without it, the cardinality $|A \cup (X \setminus A)'|$ would be the same for all $\mathcal{S}$.

For a subset of variables $A \subseteq X$, let A' and A'' be defined as $A' = \{x' \mid x \in A\}$ and $A'' = \{x'' \mid x \in A\}$. Then, the effect of the conjunct $\psi(X, Y) \to t$ in T is that, for every subset $A \subseteq X$ the following equivalence holds: The assignment I on X with $I^{-1}(1) = A$ is a model of $\varphi(X)$ if and only if $A \cup (X \setminus A)' \cup \{\psi(X, Y) \to t\} \models \{t\}$. Thus, for every $A \subseteq X$, we have the following equivalences. The assignment I on X with $I^{-1}(1) = A$ is a model of $\varphi(X)$ if and only if $A \cup (X \setminus A)' \cup A''$ is a solution of $\mathcal{P}$. Moreover, the previous assignment I is cardinality-minimal (resp. weight-minimal) if and only if $A \cup (X \setminus A)' \cup A''$ is a cardinality-minimal (resp. a weight-minimal) solution of $\mathcal{P}$. This accomplishes a parsimonious reduction to #-$\leq$-ABDUCTION (resp. #-$\sqsubseteq$-ABDUCTION). □

#-$\leq_P$-ABDUCTION with no restriction on the number of priorities requires some preparatory work. For this purpose, we first consider the appropriate version of #SAT.

Problem: #MIN-LEX–Π_kSAT
Input: A quantified Boolean formula $\varphi(X) = \forall Y_1 \exists Y_2 \cdots Q Y_k \ \psi(X, Y_1, \ldots, Y_k)$ and a subset $X' = \{x_1, \ldots, x_\ell\} \subseteq X$, such that $Q = \forall$ (resp. $Q = \exists$) and $\psi(X, Y_1, \ldots, Y_k)$ is in DNF (resp. in CNF) if k is odd (resp. k is even).
Output: Number of satisfying assignments $I \colon X \to \{0, 1\}$ of the formula $\varphi(X)$, such that $(I(x_1), \ldots, I(x_\ell))$ is lexicographically minimal.

As usual, #MIN-LEX–Π_0SAT represents the aforementioned problem for unquantified formulas, therefore we denote it as #MIN-LEX-SAT.

Theorem 2. *#MIN-LEX–Π_kSAT is $\#\cdot\mathrm{Opt}_{k+1}\mathrm{P}$-complete. In particular, #MIN-LEX-SAT is $\#\cdot\mathrm{OptP}$-complete.*

Proof. We only give the proof for #MIN-LEX-SAT, since the generalization to higher levels of the hierarchy is obvious.

In order to prove the membership, we show that #MIN-LEX-SAT can be solved by an appropriate NP-transducer M. We give a high-level description of M: It takes

as input an arbitrary propositional formula φ with variables in X plus a subset $X' = \{x_1, \ldots, x_\ell\} \subseteq X$ of distinguished variables. M non-deterministically enumerates all possible truth assignments $I\colon X \to \{0,1\}$, such that every computation path of M corresponds to exactly one assignment I. On each path, M checks in polynomial time if I is a model of φ. If this is the case, then the computation path is accepting. The output written by M on each path is the binary string $(I(x_1), \ldots, I(x_\ell))$. Finally, we define the optimal value of M to be the minimum. Obviously, the accepting paths of M outputting the optimal value correspond one-to-one to the satisfying assignments I of φ, such that $(I(x_1), \ldots, I(x_\ell))$ is lexicographically minimal.

For the hardness proof, let L be an arbitrary *minimum* problem in $\#\cdot\mathrm{OptP}$. We show that there exists a parsimonious reduction from L to #MIN-LEX-SAT. Since L is in $\#\cdot\mathrm{OptP}$, there exists an NP-transducer M for L. On input w, the transducer M produces an output of length $\leq p(|w|)$ on every branch for some polynomial p. Without loss of generality, we may assume that M actually produces an output of length exactly $= p(|w|)$. Now let w be an arbitrary instance of L and let $N = p(|w|)$ denote the length of the output on every computation path. Analogously to Cook's theorem (see [6]), there exists a propositional formula φ with variables X, such that there is a one-to-one correspondence between the satisfying truth assignment of φ and the successful computations of M on w. Moreover, X and φ can be extended in such a way that the output on each successful computation path is encoded by the variables $X' = \{x_1, \ldots, x_N\}$, i.e., for every successful computation path π, the truth values $(I(x_1), \ldots, I(x_N))$ of the corresponding model I of φ represent exactly the output on the path π. But then there is indeed a one-to-one correspondence between the computation paths of M on w, such that M outputs the minimum on these paths and the satisfying assignments of the (extended) formula φ, such that the truth values on $(x_1, \ldots, x_N)$ are lexicographically minimal. □

We also need the usual restriction of the previous problem to three literals per clause.

Problem: #MIN-LEX-3SAT
Input: A propositional formula φ in conjunctive normal form over the variables X with at most three literals per clause and a subset $X' = \{x_1, \ldots, x_\ell\} \subseteq X$.
Output: Number of satisfying assignments $I\colon X \to \{0,1\}$ of the formula φ, such that $(I(x_1), \ldots, I(x_\ell))$ is lexicographically minimal.

Since there exists a parsimonious reduction from #SAT to #3SAT (see [13]), the same reduction implies the following consequence of Theorem 2.

Corollary 1. #MIN-LEX-3SAT *is* $\#\cdot\mathrm{OptP}$*-complete.*

Theorem 3. #-$\leq_P$-ABDUCTION *without restriction on the number of priorities and* #-$\sqsubseteq_P$-ABDUCTION *with or without restriction on the number of priorities are* $\#\cdot\mathrm{Opt}_2\mathrm{P}$*-complete.* #-$\leq_P$-ABDUCTION *is* $\#\cdot\mathrm{Opt}_2\mathrm{P}[\log n]$*-complete if the number of priorities is bounded by a constant.*

Proof. For the membership proof, we slightly modify the $\Sigma_2\mathrm{P}$-transducer M from the membership proof of Theorem 1. Again, M non-deterministically enumerates all subsets $\mathcal{S} \subseteq H$, such that every computation path of M corresponds to exactly one $\mathcal{S} \subseteq H$.

By two calls to an NP-oracle, M checks on every path whether $T \cup \mathcal{S}$ is consistent (i.e., satisfiable) and whether $T \cup \mathcal{S} \models M$ holds. If both oracle calls answer "yes", then $\mathcal{S}$ is a solution of $\mathcal{P}$ and the computation path is accepting. Only the output written by M on each path has to be modified with respect to the proof of Theorem 1: Suppose that the input PAP $\mathcal{P}$ has K priorities $H_1, \ldots, H_K$. Then M computes on every computation path the vector $(c_1, \ldots, c_K)$, where c_i is the cardinality (resp. the total weight) of $\mathcal{S} \cap H_i$ for every i. Without loss of generality we may assume for every i that, on all paths, the binary representation of the numbers c_i has identical length (by adding appropriately many leading zeros). Then M simply outputs this vector $(c_1, \ldots, c_K)$, considered as a single number in binary. Finally, we again define the optimal value of M as the minimum. Obviously, the accepting paths of M outputting the optimal value correspond one-to-one to the $\leq_P$-minimal (resp. $\sqsubseteq_P$-minimal) solutions of the PAP $\mathcal{P}$. If there are no restrictions on the number K of priorities or if we consider weight-minimality, then the output of M has polynomial length. Indeed, Since $K \leq |H|$ always holds, because in the extremal case each hypothesis has its own priority class, we need at most $|H|$ bits. The length of each c_i is bounded by $\log |H|$ bits, since $c_i \leq |H|$ holds. We need $O(K \log |H|)$ bits to represent the vector $(c_1, \ldots, c_K)$. If K is constant, this becomes $O(\log |H|)$.

For the hardness part, only the $\#\cdot\mathrm{Opt}_2\mathrm{P}$-hardness of #-$\leq_P$-ABDUCTION without restriction on the number of priorities has to be shown. The remaining cases follow from the corresponding hardness result without priorities in Theorem 1. We reduce the #MIN-LEX–Π_1SAT problem to #-$\leq_P$-ABDUCTION. Let an arbitrary instance of #MIN-LEX–Π_1SAT be given by the quantified Boolean formula $\varphi(X) = \forall Y\ \psi(X, Y)$ with $X = \{x_1, \ldots, x_n\}$ and the subset $X' = \{x_1, \ldots, x_\ell\} \subseteq X$. Let t, $Q = \{q_1, \ldots, q_n\}$ $R = \{r_1, \ldots, r_\ell\}$, $Z = \{z_1, \ldots, z_n\}$, and $Z' = \{z'_1, \ldots, z'_\ell\}$ be fresh variables. Then we define the PAP $\mathcal{P} = \langle V, H, M, T \rangle$ as follows:

$$
\begin{aligned}
V &= X \cup Y \cup Z \cup Z' \cup Q \cup R \cup \{t\} \\
H &= X \cup Z \cup Z' \text{ with} \\
&\qquad H_1 = \{x_1\}, \ldots, H_\ell = \{x_\ell\}, \text{ and } H_{\ell+1} = (X \setminus X') \cup Z \cup Z' \\
M &= Q \cup R \cup \{t\} \\
T &= \{\psi(X, Y) \to t\} \cup \{\neg x_i \vee \neg z_i, x_i \to q_i, z_i \to q_i \mid 1 \leq i \leq n\} \\
&\quad \cup \{\neg z_i \vee \neg z'_i, z_i \to r_i, z'_i \to r_i \mid 1 \leq i \leq \ell\}
\end{aligned}
$$

The idea of the variables in Q, R, Z, and Z' is similar to the the variables Q, R, X', and X'' in the proof of Theorem 1. They ensure that every solution $\mathcal{S}$ of $\mathcal{P}$ contains exactly n variables out of the $2n$ variables in $H_{\ell+1}$. This can be seen as follows. By the clauses $\neg x_i \vee \neg z_i, x_i \to q_i, z_i \to q_i$ with $i \in \{1, \ldots, n\}$, every solution contains exactly one of $\{x_i, z_i\}$. Of course, the variables x_i with $i \in \{1, \ldots, \ell\}$ are not in $H_{\ell+1}$. However, the clauses $\neg z_i \vee \neg z'_i, z_i \to r_i, z'_i \to r_i$ with $i \in \{1, \ldots, \ell\}$ ensure that every solution contains exactly one of $\{z_i, z'_i\}$. In other words, for every $i \in \{1, \ldots, \ell\}$ every solution contains either $\{x_i, z'_i\}$ or $\{z_i\}$.

There is a one-to-one correspondence between the models of $\varphi(X)$ which are lexicographically minimal on X' and the $\leq_P$-minimal solutions of $\mathcal{P}$. Indeed, let I be a model of $\varphi(X)$ which is lexicographically minimal on X'. Then I can be extended to

exactly one $\leq_P$-minimal solution $\mathcal{S}$ of $\mathcal{P}$, namely $\mathcal{S} = I^{-1}(1) \cup \{z_i \mid 1 \leq i \leq n$ and $I(x_i) = 0\} \cup \{z'_i \mid 1 \leq i \leq \ell$ and $I(x_i) = 1\}$.

Conversely, let $\mathcal{S}$ be a $\leq_P$-minimal solution of $\mathcal{P}$. Then we obtain a lexicographically minimal model I of $\varphi(X)$ simply by restricting $\mathcal{S}$ to X, i.e. $I(x) = 1$ for all $x \in \mathcal{S} \cap X$ and $I(x) = 0$ otherwise. □

4 Special Cases

We consider the special cases of propositional abduction problems, where the theory is presented by Horn, definite Horn, dual Horn, or bijunctive formulas. Recall the following counting problem introduced in [10].

Problem: #MIN-CARD-VERTEX-COVER (RESP. #MIN-WEIGHT-VERTEX-COVER)
Input: Graph $G = (V, E)$ (plus a weight function $w\colon V \to \mathbb{N}$ in case of #MIN-WEIGHT-VERTEX-COVER).
Output: Number of vertex covers of G with minimal cardinality (resp. with minimal weight), i.e., cardinality-minimal (resp. weight-minimal) subsets $C \subseteq V$ such that $(u, v) \in E$ implies $u \in C$ or $v \in C$.

In [10], it was shown that #MIN-CARD-VERTEX-COVER is $\#\cdot\mathrm{OptP}[\log n]$-complete while #MIN-WEIGHT-VERTEX-COVER is $\#\cdot\mathrm{OptP}$-complete.

Theorem 4. *#-$\leq$-ABDUCTION is $\#\cdot\mathrm{OptP}[\log n]$-complete and #-$\sqsubseteq$-ABDUCTION is $\#\cdot\mathrm{OptP}$-complete for Horn, definite Horn, dual Horn, or bijunctive theories.*

Proof. For the membership part, we construct a transducer M exactly as in the proof of Theorem 1. The only difference is that we can now check in *deterministic polynomial time* whether $T \cup \mathcal{S}$ is consistent (i.e., satisfiable) and whether $T \cup \mathcal{S} \models M$ holds. Hence, we end up with the desired NP-transducer (rather than a Σ_2P-transducer) since we no longer need an NP-oracle.

The hardness is shown by a reduction from #MIN-CARD-VERTEX-COVER (resp. #MIN-WEIGHT-VERTEX-COVER). Let an arbitrary instance of #MIN-CARD-VERTEX-COVER be given by the graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. By slight abuse of notation, we consider the elements in V and E also as propositional variables and set $X = \{v_1, \ldots, v_n\}$ and $R = \{e_1, \ldots, e_m\}$. In case of #MIN-WEIGHT-VERTEX-COVER, we additionally have a weight function w defined on the variables in X. Then we define the PAP $\mathcal{P} = \langle W, H, M, T\rangle$ as follows.

$$W = X \cup R, \quad H = X, \quad M = R$$
$$T = \{v_i \to e_j \mid v_i \in e_j, 1 \leq i \leq n, 1 \leq j \leq m\}$$

The resulting theory contains only clauses which are, at the same time, Horn, definite Horn, dual Horn, and bijunctive. Obviously, for every subset $X' \subseteq X = V$ the following equivalence holds: X' is a solution of $\mathcal{P}$ if and only if X' is a vertex cover of G. But then there exists also a one-to-one correspondence between the cardinality-minimal (resp. weight-minimal) solutions of $\mathcal{P}$ and the cardinality-minimal (resp. weight-minimal) vertex covers of G. □

Again, #-$\leq_P$-ABDUCTION with no restriction on the number of priorities requires some preparatory work. For this purpose, we first consider an appropriate variant of counting the vertex covers of a graph.

Problem: #MIN-LEX-VERTEX-COVER
Input: Graph $G = (V, E)$ and a subset $V' = \{v_1, \ldots, v_\ell\} \subseteq V$.
Output: Number of vertex covers C of G, such that $(\chi(v_1), \ldots, \chi(v_\ell))$ is lexicographically minimal, where χ is the characteristic function of the vertex cover C.

Theorem 5. *#MIN-LEX-VERTEX-COVER is #·OptP-complete.*

Proof. In order to prove the membership, we show that #MIN-LEX-VERTEX-COVER can be solved by the following NP-transducer M. It takes as input an arbitrary graph $G = (V, E)$ with distinguished vertices $V' = \{v_1, \ldots, v_\ell\}$. M non-deterministically enumerates all subsets $C \subseteq V$, such that every computation path of M corresponds to exactly one such subset C. If C is a vertex cover of G, then the computation path is accepting. The output written by M on each path is the binary vector $(\chi_C(v_1), \ldots, \chi_C(v_\ell))$. Obviously, the accepting paths of M outputting the minimal value correspond one-to-one to the vertex covers C of G, such that $(\chi_C(v_1), \ldots, \chi_C(v_\ell))$ is lexicographically minimal.

The hardness proof is by a parsimonious reduction from #MIN-LEX-3SAT. In fact, this is the same reduction as in the standard NP-completeness proof of VERTEX COVER by reduction from 3SAT to VERTEX COVER, see e.g. [6]. Let $\varphi(x_1, \ldots, x_k)$ be a propositional formula in CNF with three literals per clause. We construct the graph $G = (V, E)$ as follows. For each variable x_i we construct an edge $e_i = (x_i, x_i')$. For each clause $c_i = l_i^1 \vee l_i^2 \vee l_i^3$ we construct three edges (l_i^1, l_i^2), (l_i^2, l_i^3), (l_i^3, l_i^1) forming a triangle t_i. Finally, we connect each positive literal z in the triangle t_i to its counterpart z in an edge $e_j = (z, z')$, as well as each negative literal $\neg z$ in the triangle t_i to its counterpart z'. The set of distinguished variables X' from #MIN-LEX-3SAT becomes the set of distinguished vertices V' in #MIN-LEX-VERTEX-COVER. □

Theorem 6. *#-$\leq_P$-ABDUCTION without restriction on the number of priorities and #-$\sqsubseteq_P$-ABDUCTION with or without restriction on the number of priorities are #·OptP-complete for Horn, definite Horn, dual Horn, or bijunctive theories. #-$\leq_P$-ABDUCTION for Horn, definite Horn, dual Horn, or bijunctive theories is #·OptP[log n]-complete if the number of priorities is restricted by a constant.*

Proof. For the membership part, we construct a transducer M exactly as in the proof of Theorem 3. The only difference is that we get an NP-transducer (rather than a Σ_2P-transducer) since we no longer need an NP-oracle for checking whether $T \cup \mathcal{S}$ is consistent (i.e., satisfiable) and whether $T \cup \mathcal{S} \models M$ holds.

For the hardness part, only the #·OptP-hardness of #-$\leq_P$-ABDUCTION without restriction on the number of priorities has to be shown. The remaining cases follow from the corresponding hardness result without priorities in Theorem 4. Let an arbitrary instance of #MIN-LEX-VERTEX-COVER be given by the graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$ and let $V' = \{v_1, \ldots, v_\ell\}$ with $\ell \leq n$. As in the proof of Theorem 4, we consider the elements in V and E also as propositional variables and set $X = \{v_1, \ldots, v_n\}$ and $R = \{e_1, \ldots, e_m\}$. In addition, let

$Q = \{q_{\ell+1}, \ldots, q_n\}$, and $Z = \{z_{\ell+1}, \ldots, z_n\}$ be fresh variables. Then we define the PAP $\mathcal{P} = \langle V, H, M, T \rangle$ as follows.

$$V = X \cup R \cup Q \cup Z, \quad M = R \cup Q$$
$$H = X \cup Z \text{ with } H_1 = \{v_1\}, \ldots, H_\ell = \{v_\ell\}, \text{ and } H_{\ell+1} = (X \setminus V') \cup Z$$
$$T = \{v_i \rightarrow e_j \mid v_i \in e_j, 1 \leq i \leq n, 1 \leq j \leq m\} \cup$$
$$\{v_i \rightarrow q_i, z_i \rightarrow q_i \mid \ell + 1 \leq i \leq n\}$$

The resulting theory contains only clauses which are, at the same time, Horn, definite Horn, dual Horn, and bijunctive. The variables Q and Z realize the familiar idea that in every $\leq_P$-minimal solution $\mathcal{S}$ of $\mathcal{P}$, for every $i \in \{\ell + 1, \ldots, n\}$, exactly one of v_i and z_i is contained in $\mathcal{S}$. It can then be easily shown that there is a one-to-one correspondence between the lexicographically minimal vertex covers of G and the $\leq_P$-minimal solutions of $\mathcal{P}$. □

5 Conclusion

In this paper, we have completed the analysis of the counting complexity of propositional abduction. Together with previous results presented in [9], we have thus achieved a full picture. Recall from [19] that counting problems may display a significantly different complexity behavior from the corresponding decision problems. Hence, the complexity of a class of problems is better understood when we analyse the counting complexity in addition to the decision complexity. By complementing the complexity results of Eiter and Gottlob [2] on decision problems related to propositional abduction with our counting complexity results in Table 1, we have thus arrived at a better understanding of the complexity of propositional abduction in various settings.

From a complexity theoretic point of view, there is another interesting aspect to the counting complexity results shown here. The class #P has been studied intensively and many completeness results for this class can be found in the literature. In contrast, for the higher counting complexity classes $\#\cdot\Pi_k\mathrm{P}$, $\#\cdot\mathrm{Opt}_k\mathrm{P}[\log n]$, and $\#\cdot\mathrm{Opt}_k\mathrm{P}$ (with $k \geq 1$) very few problems had been shown to be complete. Our results on the counting complexity of propositional abduction thus also lead to a better understanding of these counting complexity classes.

For future work, we plan to extend the complexity analysis of many more families of decision problems in the artificial intelligence domain (like, e.g., closed-world reasoning in various settings) to counting problems. Moreover, we would also like to extend the abduction cases studied in this paper to yet another case, namely the case of affine theories, i.e.: the theory T is an affine system $AX = b$ over $\mathbb{Z}_2$. This case was in fact dealt with in [9] for #-$\preceq$-abduction with $\preceq \in \{=, \subseteq, \subseteq_P\}$. There are obvious upper and lower bounds also for #-$\preceq$-abduction with affine theories when the preorder $\preceq$ is in $\{\leq, \sqsubseteq, \leq_P, \sqsubseteq_P\}$. However, proving tight complexity bounds also for these cases has to be left as an open problem for future work.

References

1. Durand, A., Hermann, M., Kolaitis, P.G.: Subtractive reductions and complete problems for counting complexity classes. Theoretical Computer Science 340(3), 496–513 (2005)
2. Eiter, T., Gottlob, G.: The complexity of logic-based abduction. Journal of the Association for Computing Machinery 42(1), 3–42 (1995)
3. Eiter, T., Makino, K.: On computing all abductive explanations. In: Proc. 18th AAAI, Edmonton, Alberta, Canada, pp. 62–67. AAAI Press, Menlo Park (2002)
4. Eiter, T., Makino, K.: Generating all abductive explanations for queries on propositional Horn theories. In: Baaz, M., Makowsky, J.A. (eds.) CSL 2003. LNCS, vol. 2803, pp. 197–211. Springer, Heidelberg (2003)
5. Fagin, R., Ullman, J.D., Vardi, M.Y.: On the semantics of updates in databases. In: Proc. 2nd PODS, Atlanta, Georgia, USA, pp. 352–365. ACM Press, New York (1983)
6. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman and Co., New York (1979)
7. Gasarch, W.I., Krentel, M.W., Rappoport, K.J.: OptP as the normal behavior of NP-complete problems. Mathematical Systems Theory 28(6), 487–514 (1995)
8. Hemaspaandra, L.A., Vollmer, H.: The satanic notations: Counting classes beyond #P and other definitional adventures. SIGACT News 26(1), 2–13 (1995)
9. Hermann, M., Pichler, R.: Counting complexity of propositional abduction. In: Veloso, M.M. (ed.) Proc. 20th IJCAI 2007, Hyderabad, India, pp. 417–422. AAAI Press, Menlo Park (2007)
10. Hermann, M., Pichler, R.: Complexity of counting the optimal solutions. In: Hu, X., Wang, J. (eds.) COCOON 2008. LNCS, vol. 5092, pp. 149–159. Springer, Heidelberg (2008)
11. Herzig, A., Lang, J., Marquis, P., Polacsek, T.: Updates, actions, and planning. In: Nebel, B. (ed.) Proc. 17th IJCAI 2001, Seattle, Washington, USA, pp. 119–124. Morgan Kaufmann, San Francisco (2001)
12. Kakas, A.C., Mancarella, P.: Database updates through abduction. In: McLeod, D., Sacks-Davis, R., Schek, H.-J. (eds.) Proc. 16th VLDB, Brisbane, Queensland, Australia, pp. 650–661. Morgan Kaufmann, San Francisco (1990)
13. Kozen, D.C.: The design and analysis of algorithms. In: Counting problems and #P, ch. 26, pp. 138–143. Springer, Heidelberg (1992)
14. Krentel, M.W.: The complexity of optimization problems. Journal of Computer and System Sciences 36(3), 490–509 (1988)
15. Krentel, M.W.: Generalizations of OptP to the polynomial hierarchy. Theoretical Computer Science 97(2), 183–198 (1992)
16. Papatheodorou, I., Kakas, A.C., Sergot, M.J.: Inference of gene relations from microarray data by abduction. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 389–393. Springer, Heidelberg (2005)
17. Peng, Y., Reggia, J.A.: Abductive inference models for diagnostic problem solving. Springer, Heidelberg (1990)
18. Poole, D.: Probabilistic Horn abduction and bayesian networks. Artificial Intelligence 64(1), 81–129 (1993)
19. Valiant, L.G.: The complexity of computing the permanent. Theoretical Computer Science 8(2), 189–201 (1979)
20. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM Journal on Computing 8(3), 410–421 (1979)

Uniform Interpolation by Resolution in Modal Logic

Andreas Herzig and Jérôme Mengin

Institut de Recherche en Informatique de Toulouse
{herzig,mengin}@irit.fr

Abstract. The problem of computing a uniform interpolant of a given formula on a sublanguage is known in Artificial Intelligence as variable forgetting. In propositional logic, there are well known methods for performing variable forgetting. Variable forgetting is more involved in modal logics, because one must forget a variable not in one world, but in several worlds. It has been shown that modal logic K has the uniform interpolation property, and a method has recently been proposed for forgetting variables in a modal formula (of mu-calculus) given in disjunctive normal form. However, there are cases where information comes naturally in a more conjunctive form. In this paper, we propose a method, based on an extension of resolution to modal logics, to perform variable forgetting for formulae in conjunctive normal form, in the modal logic K.

1 Introduction

An interpolant of logical formulae ϕ and ψ such that $\phi \models \psi$ in a logic L is a formula χ that contains only variables that appear in both ϕ and ψ, and such that $\phi \models \chi$ and $\chi \models \psi$. A uniform interpolant of ϕ with respect to a sublanguage $\mathcal{L}'$ of the language of ϕ is a formula $\chi \in \mathcal{L}'$ entailed by ϕ that can act as an interpolant for any $\psi \in \mathcal{L}'$: if $\phi \models \psi$, then $\chi \models \psi$. In other words, χ behaves like ϕ when $\mathcal{L}'$ is concerned, in the sense that ψ has the same $\mathcal{L}'$-logical consequences as ϕ [1].

When the language $\mathcal{L}'$ is defined as the set of formulae that contain no variable of a given set P, the problem of computing a uniform interpolant of ϕ on $\mathcal{L}'$ is known in Artificial Intelligence as *variable forgetting*. In propositional logic, there are well known methods for performing variable forgetting [2,3].

Variable forgetting is more involved in modal logics, because one must forget a variable not in one world, but in several worlds. It has been shown that modal logic K has the uniform interpolation property [4,5] (examples of logics that do not have the uniform interpolation property include classical first order logic and S4 [6]).

[1] propose a simple method for forgetting variables in a modal formula (of μ-calculus) given in disjunctive normal form.[7,8] propose methods to construct modal interpolants for general modal formulas, based on sequent calculus or tableaux methods; both methods work by implicitly decomposing the formula in disjunctive normal form first. However, there are cases where information comes naturally in a more conjunctive form. In this paper, we propose a method, based on an extension of resolution to modal logics, to perform variable forgetting for formulae in conjunctive normal form, in the modal logic K.

In the next section, we briefly recall the syntax and semantics of K, and the definitions of disjunctive and conjunctive normal forms in this logic. In section 3, we recall

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 219–231, 2008.

Enjalbert and Fariñas' resolution system for K. In section 4, we explain how their resolution system can be used to perform variable forgetting in that logic.

2 The Logic K

The language of K, over a set of propositional variables $\mathcal{P}$, is the smallest set $\mathcal{L}$ of formulae that contains $\mathcal{P}$ and is closed under conjunction $\wedge$, negation $\neg$ and necessity. $\Box$

A *pointed model* for K is a tuple $m = (W, R, I, w)$, where W is a non-empty set, R is a binary relation over W, I assigns to every $p \in \mathcal{P}$ and every $w' \in W$ a value $I(p, w') \in \{\mathsf{true}, \mathsf{false}\}$.

Satisfaction of formula ϕ by model $m = (W, R, I, w)$ is defined by induction as follows:

- $m \models p$ if and only if $I(p, w) = \mathsf{true}$, for every $p \in \mathcal{P}$;
- $m \models \neg\phi$ if and only if $m \not\models \phi$;
- $m \models \phi \wedge \psi$ if and only if $m \models \phi$ and $m \models \psi$;
- $m \models \Box\phi$ if and only if $(W, R, I, w') \models \phi$ for every $w' \in W$ such that wRw'.

A set of formulae S is said to entail formula ψ, written $S \models \psi$, if for every pointed model m that satisfies every formula in S, m satisfies ψ too. In this case, we will also say that ψ is a logical consequence of S. We will write $S \not\models \psi$ when this is not the case. Given a finite set of formulae $\{\phi_1, \ldots, \phi_n\}$, we will sometimes write $\phi_1, \ldots, \phi_n \models \psi$ instead of $\{\phi_1, \ldots, \phi_n\} \models \psi$.

As usual, two other connectors $\vee$ and $\Diamond$ are introduced and defined as abbreviations: $\phi \vee \psi \equiv \neg(\neg\phi \wedge \neg\psi)$ and $\Diamond\phi \equiv \neg\Box\neg\phi$.

For that local definition of logical consequence, the deduction theorem holds for finite S: $S \models \phi$ if and only if $\{\} \models \neg \bigvee S \vee \phi$.

There are various ways to define conjunctive and disjunctive normal forms in modal logic (see e.g. [9,1,10]). In this paper, we will consider normal forms close to the ones defined by Enjalbert and Fariñas. Literals, clauses and terms can be defined using a grammar in BNF:

$$\begin{aligned}
\mathsf{Lit}C &::= p \mid \neg p \mid \Box\mathsf{Clause} \mid \Diamond\mathsf{CNF} \\
\mathsf{Lit}T &::= p \mid \neg p \mid \Box\mathsf{DNF} \mid \Diamond\mathsf{Term} \\
\mathsf{Clause} &::= \mathsf{Lit}C \mid \mathsf{Clause} \vee \mathsf{Clause} \mid \bot \\
\mathsf{Term} &::= \mathsf{Lit}T \mid \mathsf{Term} \wedge \mathsf{Term} \mid \top \\
\mathsf{CNF} &::= \mathsf{Clause} \mid \mathsf{CNF}, \mathsf{CNF} \\
\mathsf{DNF} &::= \mathsf{Term} \mid \mathsf{DNF} \vee \mathsf{DNF}
\end{aligned}$$

According to this definition, clauses are disjunctions of literals of type "C" (for "Clause"): such a literal is either a propositional literal, or a clause behind a $\Box$, or a set/conjunction of clauses behind a $\Diamond$. A CNF is then a set/conjunction of clauses. This corresponds exactly to the definition of clauses used by Enjalbert and Fariñas. A term is the dual of a clause: it is a conjunction of literals of type "T", where such a literal is either a propositional literal,

or a term behind a $\Diamond$, or a disjunction of terms behind a $\Box$. Every formula of K has an equivalent DNF and an equivalent CNF.

Uniformly interpolating a DNF is relatively simple, because of two properties of interpolation. The first one is general in extensions of classical propositional logic: if χ and χ' are respective uniform interpolants of ϕ and ϕ' (on a same language), then $\chi \vee \chi'$ is a (uniform) interpolant of $\phi \vee \phi'$ (since if $\phi \vee \phi' \models \psi$, then $\phi \models \psi$ and $\phi' \models \psi$, thus $\chi \models \psi$ and $\chi \models \psi'$). The second one concerns the interplay of modal connectives and interpolation: a uniform interpolant of $\phi \wedge \Box\phi' \wedge \Diamond\phi_1 \wedge \ldots \Diamond\phi_n$, where ϕ does not contain any modality, is simply $\chi \wedge \Box\chi' \wedge \Diamond(\chi_1 \wedge \chi)' \wedge \ldots \wedge \Diamond(\chi_n \wedge \chi)'$, where the $'$s indicate interpolants of the formulas to which they are applied (this is equivalent to the result proved in [1], and used in [11], on interpolation for slightly different disjunctive normal forms: a uniform interpolant of $\phi \wedge \Diamond\phi_1 \wedge \ldots \wedge \Diamond\phi_n \wedge \Box(\phi_1 \vee \ldots \vee \phi_n)$ is $\phi' \wedge \Diamond\phi'_1 \wedge \ldots \wedge \Diamond\phi'_n \wedge \Box(\phi'_1 \vee \ldots \vee \phi'_n)$).

In the sequel, we propose a method to compute a uniform interpolant of a CNF.

3 Resolution in Modal Logic

From a model theoretic point of view, resolution can be understood as an application of the general set theoretic property:

$$(M \cup N) \cap (M' \cup N') \subseteq (M \cap M') \cup N \cup N'.$$

In terms of logical formulae, this can be rephrased as:

$$\phi \vee C, \psi \vee C' \models (\phi \wedge \psi) \vee C \vee C'.$$

In the case where $\psi = \neg\phi$, we obtain the usual resolution rule of classical logic: $\phi \vee C, \neg\phi \vee C' \models C \vee C'$. But in modal logic, this can be used to produce other inference rules. In particular, if $\phi \wedge \psi \models \chi$, then $\Diamond\phi, \Box\psi \models \Diamond\chi$ and $\Box\phi, \Box\psi \models \Box\chi$, thus

$$\Diamond\phi \vee C, \Box\psi \vee C' \models \Diamond\chi \vee C \vee C'$$
$$\Box\phi \vee C, \Box\psi \vee C' \models \Box\chi \vee C \vee C'$$

Note that these deductions can be made at any depth in a modal formula in negation normal form, since if $\phi \models \chi$, then $\Box\phi \models \Box\chi$, and $\Diamond(\phi \wedge \psi) \models \Diamond(\phi \wedge \psi \wedge \chi)$.

In order to define a practical inference system, one has to define precisely which inferences are allowed. Enjalbert and Fariñas [10] define a set of inference rules that we recall below. In the sequel, we adopt an *in line* notation for inference rules: $A_1, \ldots, A_n \Longrightarrow B$ denotes an inference rule whose premisses are $A_1, \ldots, A_n$, and whose consequent is B. Enjalbert and Fariñas' resolution system for K is defined by a set of conditional, recursive "meta"-rules; it is the smallest set of inference rules that contains, for $\alpha \in \mathcal{P} \cup \{\bot\}$:[1]

[1] The "meta"-system of [10] is slightly different from the presentation given here, mainly because we do not separate the introduction of $\vee$ from the introduction of modalities or of $\bot$.

- (rule $\bot$) $\boxed{\Box\bot} \vee C_1', \boxed{\Diamond E} \vee C_2' \Longrightarrow_\bot C_1' \vee C_2'$;
- (rule p) $\boxed{p} \vee C_1', \boxed{\neg p} \vee C_2' \Longrightarrow_p C_1' \vee C_2'$;
- (rule $\Box\Diamond$) $\boxed{\Box C_1} \vee C_1', \boxed{\Diamond(C_2, E)} \vee C_2' \Longrightarrow_\alpha \Diamond(C_2, E, C_3) \vee C_1' \vee C_2'$ if $C_1, C_2 \Longrightarrow_\alpha C_3$;
- (rule $\Box\Box$) $\boxed{\Box C_1} \vee C_1', \boxed{\Box C_2} \vee C_2' \Longrightarrow_\alpha \Box C_3 \vee C_1' \vee C_2'$ if $C_1, C_2 \Longrightarrow_\alpha C_3$;
- (rule $\Diamond 2$) $\boxed{\Diamond(C_1, C_2, E)} \vee C \Longrightarrow_\alpha \Diamond(C_1, C_2, E, C_3) \vee C$ if $C_1, C_2 \Longrightarrow_\alpha C_3$;
- (rule $\Diamond 1$) $\boxed{\Diamond(C_1, E)} \vee C \Longrightarrow_\alpha \Diamond(C_1, E, C_2) \vee C$ if $C_1 \Longrightarrow_\alpha C_2$;
- (rule $\Box$) $\boxed{\Box C_1} \vee C \Longrightarrow_\alpha \Box C_2 \vee C$ if $C_1 \Longrightarrow_\alpha C_2$;

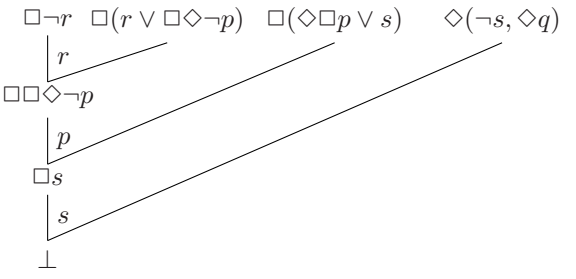
We assume that the consequents are always *normalized*, that is simplified using the following equivalences: $A \vee A \vee B \equiv A \vee B$, $A \vee \bot \equiv A$, $\Diamond\bot \equiv \bot$, $A \wedge \bot \equiv \bot$. Hence $\Box p, \Diamond\neg p \Rightarrow \bot$ because $\Box p, \Diamond\neg p \Rightarrow \Diamond\bot$ is inferred from the axiom (rule p): $p, \neg p \Rightarrow \bot$ by (rule $\Box, \Diamond$), and then $\Diamond\bot$ is normalized to $\bot$. In the description above, we have drawn a box around, in each premiss, the literal *resolved upon*. Let us stress that each of the inference rules of this resolution system is "meta"-derived from a unique simple (rule $\bot$) or (rule p) for some $p \in \mathcal{P}$; if it is derived from some rule $\neg p$, we call p the resolved variable. The α that indexes the rules denotes this variable or $\bot$. The *height* of the derivation of a resolution rule will be the number of meta-rules used to obtain it, minus 1.

Example 1. The following are derivations of resolution rules:

- $$\dfrac{\dfrac{\neg r, r \vee \Box\Diamond\neg p \Longrightarrow_r \Box\Diamond\neg p \quad \text{(rule } r\text{)}}{\Box\neg r, \Box(r \vee \Box\Diamond\neg p) \Longrightarrow_r \Box\Box\Diamond\neg p} \quad \text{(rule } \Box\Box\text{)}}$$
  (This derivation is of height 1.)
- $$\dfrac{\dfrac{\dfrac{\neg p, p \Longrightarrow_p \bot \quad \text{(rule } p\text{)}}{\Diamond\neg p, \Box p \Longrightarrow_p \bot} \quad \text{(rule } \Box\Diamond\text{)}}{\Box\Diamond\neg p, \Diamond\Box p \vee s \Longrightarrow_p s} \quad \text{(rule } \Box\Diamond\text{)}}{\Box\Box\Diamond\neg p, \Box(\Diamond\Box p \vee s) \Longrightarrow_p \Box s} \quad \text{(rule } \Box\Box\text{)}}$$
  (This derivation is of height 3.)

Definition 1. *A* deduction by resolution *of clause C from clause set S is a sequence of inferences by resolution $I_1, \ldots, I_n$ such that for every I_i, each premises of I_i belongs to S, or is the consequent of rule I_j for some $j < i$.*

Example 2. Let $S = \{\Box\neg r, \Box(r \vee \Box\Diamond\neg p), \Box(\Diamond\Box p \vee s), \Diamond(\neg s, \Diamond q)\}$. There is a deduction by resolution of $\bot$ from S, for example with the rules $I_1 = \Box\neg r, \Box(r \vee \Box\Diamond\neg p) \Longrightarrow_r \Box\Box\Diamond\neg p$, $I_2 = \Box\Box\Diamond\neg p, \Box(\Diamond\Box p \vee s) \Longrightarrow_p \Box s$, and $I_3 = \Box s, \Diamond(\neg s, \Diamond q) \Longrightarrow_s \bot$:



Enjalbert and Fariñas show that their resolution system is sound and complete for K with respect to refutation: a given clause set S is unsatisfiable ($S \models \bot$) if and only if there is a deduction by resolution of $\bot$ from some subset of S. Moreover, since the modal depth of the consequent of some inference by resolution cannot be greater that the depth of its premises, there can be only a finite set of clauses that can be deduced by resolution from a finite set of clauses S (remember that redundant literals are implicitly simplified when resolutions are performed).

We will show in the next section that forgetting variables from P in a set of modal clauses S can be done by performing all possible resolutions on variables from P, and then eliminating all occurrences of the variables of P. The property of resolution that will enable us to do that is that, given a subset of propositional variables P, we can re-arrange inferences so that resolutions on variables from P appear before other resolutions. Formally:

Proposition 1. *Given a subset of propositional variables $P \subseteq \mathcal{P}$, if there is a deduction by resolution of clause C from clause set S, then there is a deduction by resolution $I_1, \ldots, I_n$ of some clause C^* from S such that C^* subsumes C and for every i, j, if I_i resolves on a variable from P whereas I_j resolves on $\bot$ or a variable from $\mathcal{P} - P$, then $i < j$.*

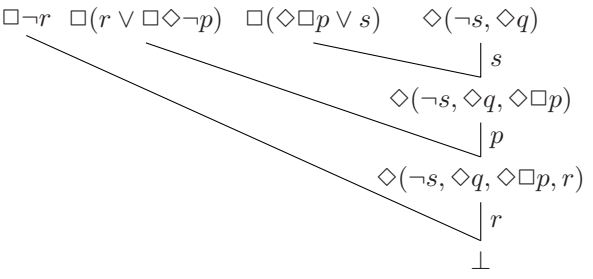
Note that the resulting clause C^* may not exactly be the original one, it may in fact be stronger. More precisely, we define subsumption as follows:

Definition 2. *A clause C subsumes a clause D if every literal of C subsumes some literal of D, and a set of clauses E subsumes a set of clauses F if every clause of F is subsumed by some clause of E. A propositional literal subsumes itself; a literal $\Box C$ subsumes a literal $\Box D$ if C subsumes D; and a literal $\Diamond E$ subsumes $\Diamond F$ if E subsumes F.*

Example 3. $\Box e \vee \Diamond(r, s, p) \vee q$ subsumes $p \vee \Box(e \vee f) \vee \Diamond(r \vee t, s) \vee q$ because:

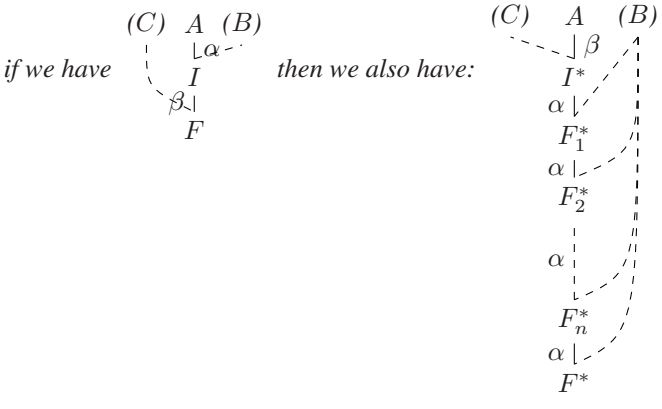
- $\Box e$ subsumes $\Box(e \vee f)$ because e subsumes $e \vee f$;
- $\Diamond(r, s, p)$ subsumes $\Diamond(r \vee t, s)$ because $r \vee t$ is subsumed by r and s is subsumed by s.
- q subsumes q

Example 2 (continued). We can re-arrange resolutions, so that resolutions on r come last:



Prop. 1 is an easy consequence of the following lemma, which we prove in appendix:

Lemma 1. *Given clauses A, B, C, I and F, if there is an α-resolution from A, possibly using side clause B, giving clause I, and a β-resolution from I, possibly using side clause C, giving clause F, then there exist clauses I^*, F_1^*, …, F_n^* (for some $n \geq 0$) and F^* such that F^* subsumes F and there is a β-resolution from A, possibly using side clause C, giving clause I^*, and a sequence of α-resolutions from I^*, possibly using side clause B, giving successively F_1^*, …, F_n^*, F^*. In other words, if $A(,B) \Longrightarrow_\alpha I$ and $I(,C) \Longrightarrow_\beta F$, then $A(,C) \Longrightarrow_\beta I^*$, and $I^*(,B) \Longrightarrow_\alpha F_1^*$ and $F_1^*(,B) \Longrightarrow_\alpha F_2^*$ and … and $F_n^*(,B) \Longrightarrow_\alpha F^*$. Or, with some pictures:*



4 Uniform Interpolation by Resolution

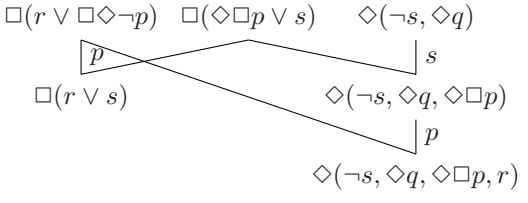
Suppose now that we want to compute a uniform interpolant for clause set S and a sublanguage $\mathcal{L}^{\mathsf{no}P}$ defined by a subset of variables P: $\mathcal{L}^{\mathsf{no}P}$ is the sublangage of $\mathcal{L}$ whose formulas have no occurrences of atoms in P. We can proceed as follows:

1. Recursively add to S all clauses obtained from S by resolutions on variables from P: this gives a set of clauses $S^{\mathsf{res}(P)}$; then
2. Suppress from $S^{\mathsf{res}(P)}$ all information about variables from P; formally, we define an operator Supp such that $\mathsf{Supp}(P, C)$ associates to clause C a clause that "forgets" what C says about variables from P:
   - if C is of the form $p \vee C'$ or $\neg p \vee C'$ for some $p \in P$, then $\mathsf{Supp}(P, C) = \top$;

– otherwise, that is, if no variable of P appears at "ground" level in C: $\mathsf{Supp}(P, C' \vee \Diamond R_1 \vee \ldots \vee \Diamond R_n \vee \Box C_1 \vee \ldots \vee \Box C_n) = C' \vee \Diamond\mathsf{Supp}(P, R_1) \vee \ldots \vee \Diamond\mathsf{Supp}(P, R_n) \vee \Box\mathsf{Supp}(P, C_1) \vee \ldots \vee \Box\mathsf{Supp}(P, C_n)$ where for each i, $\mathsf{Supp}(P, R_i) = \{\mathsf{Supp}(P, C) \mid C \in R_i\}$. We also perform the natural simplification: $\Box\top \equiv C \vee \top \equiv \top$ and $R \cup \{\top\} \equiv R$.

Let $S^{\mathsf{no}P} = \{\mathsf{Supp}(P, C) \mid C \in S^{\mathsf{res}(P)}\}$. We claim that $S^{\mathsf{no}P}$ is a uniform interpolant of S on $\mathcal{L}^{\mathsf{no}P}$. In order to see this, suppose that ϕ is a formula of $\mathcal{L}^{\mathsf{no}P}$ such that $S \models \phi$: let S' be a conjunctive normal form of $\neg\phi$, by refutation completness of Enjalbert and Fariñas' resolution system, there is a deduction by resolution of $\bot$ from $S \cup S'$. Let us use Prop. 1 and re-arrange this deduction to get a sequence of inferences where all resolutions on variables from P are before the others: it has the form $I_1, \ldots I_k, \ldots, I_n$, where inferences $I_1, \ldots, I_k$ are on variables from P, and $I_{k+1}, \ldots, I_n$ are resolutions on $\bot$ or on other variables. Let $C_1, \ldots, C_m$ be the premisses of the inferences $I_{k+1}, \ldots, I_n$ that are in S or that are consequents of inferences $I_1, \ldots, I_k$; and let R be the set of clauses obtained by applying $\mathsf{Supp}(P, \bullet)$ to $C_1, \ldots, C_m$: then $R \subseteq S^{\mathsf{no}P}$, and there is a deduction by resolution from $R \cup S'$ for $\bot$, since there is a deduction by resolution from $\{C_1, \ldots, C_m\} \cup S'$ and clauses in R are "simpler" than $C_1, \ldots, C_m$. Thus $S^{\mathsf{no}P} \cup S' \models \bot$, hence $S^{\mathsf{no}P} \models \phi$.

Example 2 (continued). Suppose we need to compute a $\mathcal{L}^{\mathsf{no}P}$ interpolant of $R = \{\Box(r \vee \Box\Diamond\neg p), \Box(\Diamond\Box p \vee s), \Diamond(\neg s, \Diamond q)\}$ where $P = \{p, q, s\}$. We first perform all possible resolutions on p, q, s:



We now compute $\mathsf{Supp}(P, R)$. The clauses $\Diamond(\neg s, \Diamond q)$, $\Diamond(\neg s, \Diamond q, \Diamond\Box p)$ and $\Box(\Diamond\Box p \vee s)$ are suppressed by $\mathsf{Supp}(P, \bullet)$ because they contain only occurrences of s p, and q. The clauses $\Box(r \vee \Box\Diamond\neg p)$ and $\Box(r \vee s)$ are discarded too, because they do not specify in which case r is true. In the clause $\Diamond(\neg s, \Diamond q, \Diamond\Box p, r)$, one conjunct is kept, r. So $R^{\mathsf{no}P} = \{\Diamond r\}$.

5 Discussion

The results above show that, although it is more complicated than in propositional logic, resolution can be used to compute a uniform interpolant in modal logic K. Note that resolution works with clauses, thus formulas have to be put in conjunctive normal form first, a step which has exponential worst-case complexity. In fact, for formulas in a rather disjunctive form, methods of [1,7,8] would be preferable, since they work best on formulas in Disjunctive Normal Form. The method of [1] supposes that the formula to be interpolated is actually in DNF. [7] proposes an operator to compute a pre-interpolant

of a modal sequent. In order to give an idea of how the method works, suppose Γ is a set of modal formulas: [7] defines an operator A_p, such that $A_p(\Gamma)$ is a formula that contains only variables of Γ, except p, which is inconsistent with Γ and such that for every ϕ such that ϕ does not contain p and $\Gamma \models \phi$, $\models A_p(\Gamma) \vee \phi$; so $\neg A_p(\Gamma)$ is a uniform interpolant of Γ (in fact, it is a post-interpolant of Γ, whereas $A_p(\Gamma)$ would be a pre-interpolant of the sequent $\Gamma \vdash \bot$). The definition of A_p is inductive. For the sake of simplicity, let us assume that the formulas in Γ are in negation normal form (all negations appear only before propositional variables). Then:

$$
\begin{array}{ll}
1 & \neg A_p(\Gamma', C_1 \wedge C_2) = \neg A_p(\Gamma', C_1, C_2) \\
2 & \neg A_p(\Gamma', C_1 \vee C_2) = \neg A_p(\Gamma', C_1) \vee \neg A_p(\Gamma', C_2) \\
3 & \neg A_p(p, \neg p, \ldots) = \bot \\
4 & \neg A_p(l_1, \ldots, l_m, \Box\Gamma_1, \ldots, \Box\Gamma_n, \Diamond\Gamma'_1, \ldots, \Diamond\Gamma'_{n'}) \\
 & \quad = \bigwedge_{l_i \neq p, \neg p} l_i \wedge \bigwedge_i \Diamond \neg A_p(\Gamma_1, \ldots, \Gamma_n, \Gamma'_i) \wedge \Box \neg A_p(\Gamma_1, \ldots, \Gamma_n)
\end{array}
$$

Here, rules 1 and 2 are used to decompose Γ in disjunctive normal form. When a conjunction contains both p and $\neg p$, it is marked inconsistent with rule 3. Finally, rule 4 is applied when Γ cannot be decomposed further, and when it does not contain opposite literals: we keep all propositional literals of Γ that do not involve p, interpolate the conjunction of the $\Box$ literals, and interpolate each $\Diamond$ literal with the conjunction of the $\Box$ literals: this is similar to the approach of [1]. The method of [8] works by constructing a tableau for a (single) formula Γ to be interpolated: essentially, all leaves that close on $p, \neg p$ are replaced by $\bot$, and all occurrences of p are removed from all other leaves; it is then possible to construct "backwards" (starting from the leaves) a new, closed tableau for a uniform interpolant of Γ. Again, the tableau construction implicitly decomposes the formula in disjunctive normal form.

However, conjunctive normal forms do occur naturally in many situations, for example when one must represent a number of properties of a given system: each property can be described by a formula, and the conjunction of theses formulas will describe the system. In this case, it is the transformation into a DNF which will be exponential. Consider for example the following conjunctive formula: $(p \vee C_1) \wedge \ldots (p \vee C_m) \wedge (\neg p \vee C'_1) \wedge \ldots (\neg p \vee C'_n)$. There are $m \times n$ resolutions on variable p that lead to $m \times n$ clauses. If we were to put the formula in disjunctive form, we would have 2^{m+n} disjuncts, of which $2^n + 2^m - 1$ do not contain both p and $\neg p$ (assuming p does not appear in the C_is nor in the C'_is).

In order to have an effective algorithm for computing the interpolant using resolution, one would need to precise a procedure to recursively compute P-resolvants of a given set of clauses. An algorithm like [12]'s saturation by set could be used, coupled with the elimination of subsumed clauses. From an implementation point of view, an important difference between resolution in propositional logic and in modal logic is the representation of clauses: they can be efficiently represented in a table when there are no modalities. From this point of view, the approach of [13] seems promising: they "flatten" modal formulas, using a naming scheme for the possible worlds, a little like skolemization in first-order logic, and then perform resolution on flat clauses. In order

to properly use this approach for interpolation, one would need to define a sort of "de-skolemization" in order to regain modal formulas after interpolation has been performed on flat clauses.

Acknowledgments

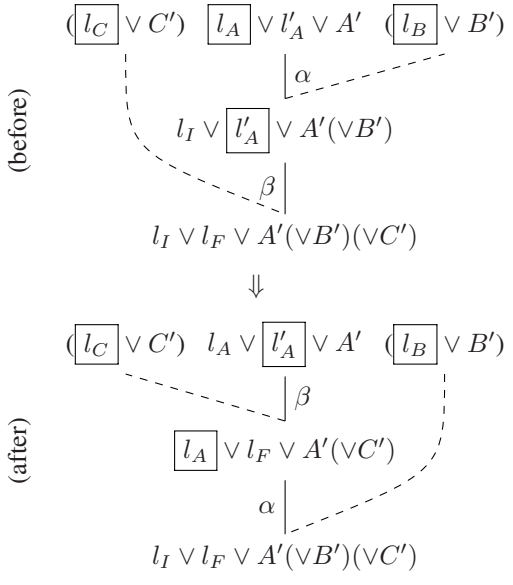
We thank Meghyn Bienvenu for many critiqual discussions on the topic of the paper, and three anonymous referees for helpful comments on an earlier version of the paper.

References

1. D'Agostino, G., Lenzi, G.: On modal mu-calculus with explicit interpolants. Journal of Applied Logic 4(3), 256–278 (2006)
2. Kohlas, J., Moral, S., Haenni, R.: Propositional Information Systems. Journal of Logic and Computation 9(5), 651–681 (1999)
3. Lang, J., Marquis, P.: Resolving inconsistencies by variable forgetting. In: Fensel, D., Giunchiglia, F., McGuinness, D.L., Williams, M.A. (eds.) Proceedings of the Eights International Conference on Principles of Knowledge Representation and Reasoning (KR 2002), pp. 239–250. Morgan Kaufmann, San Francisco (2002)
4. Visser, A.: Bisimulations, model descriptions and propositional quantifiers. In: Hájek, P. (ed.) Gödel 1996: Logical foundations of mathematics, computer science and physics – Kurt Gödel's legacy. A K Peters Ltd (2001)
5. Ghilardi, S., Zawadowski, M.W.: Undefinability of propositional quantifiers in the modal system s4. Studia Logica 55(2), 259–271 (1995)
6. Ghilardi, S., Zawadowski, M.W.: Sheaves, Games, and Model Completions. Trends in Logic, vol. 14. Kluwer, Dordrecht (2002)
7. Bílková, M.: Uniform interpolation and propositional quantifiers in modal logics. Studia Logica 85, 1–31 (2007)
8. Kracht, M.: Modal consequence relations. In: Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic. Elsevier, Amsterdam (2006)
9. Bienvenu, M.: Prime Implicates and Prime Implicants in Modal Logic. In: Holte, R.C., Howe, A. (eds.) Proceedings of the 22nd National Conference on Artifcial Intelligence (AAAI 2007), pp. 379–384. AAAI Press, Menlo Park (2007)
10. Enjalbert, P., del Cerro, L.F.: Modal resolution in clausal form. Theoretical Computer Science 65(1), 1–33 (1989)
11. ten Cate, B., Conradie, W., Marx, M., Venema, Y.: Definitorially complete description logics. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 79–89. AAAI Press, Menlo Park (2006)
12. Besnard, P., Quiniou, R., Quinton, P.: A theorem prover for a decidable subset of default logic. In: Proceedings of the Third National Conference on Artificial Intelligence, pp. 27–30. Morgan Kaufmann, San Francisco (1983)
13. Areces, C., de Rijke, M., de Nivelle, H.: Resolution in modal, description and hybrid logic. Journal of Logic and Computation 11(5), 717–736 (2001)

Appendix: Proof of Lemma 1

The proof is by induction on the height of the derivation height of the α-resolution. The base case corresponds to an α-resolution whose derivation height is zero: these are the propositional resolutions and the $\Box\bot$ resolution. These are particular cases of a more general case, where the second resolution is not on the literal obtained from the first resolution (when the α resolution is a propositional one or a $\Box\bot$, it produces no literal). Then A is of the form $A = l_A \vee l'_A \vee A'$, whereas $B = l_B \vee B'$ and $C = l_C \vee C'$ (if needed), with $l_A(, l_B) \Longrightarrow_\alpha l_I$, and $l'_A(, l_C) \Longrightarrow_\beta l_F$. Then we just have to take $I = l_A \vee l_F \vee A'(\vee C')$ and $F^* = F$. Pictorially:

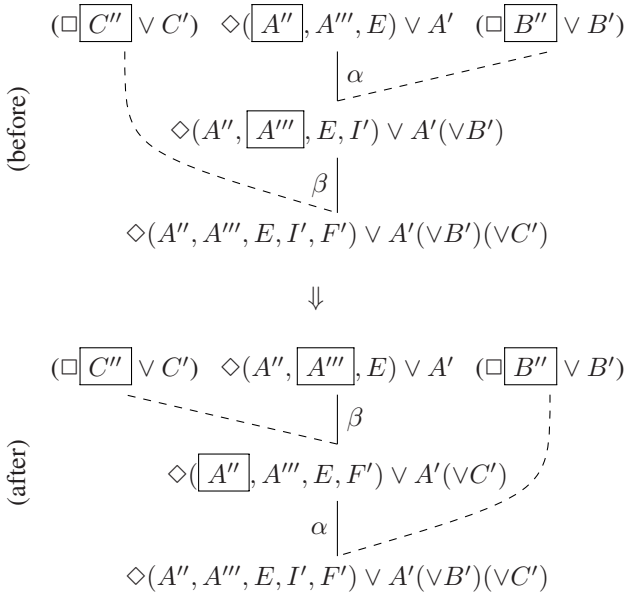


Suppose now that the result holds when the derivation height of the α resolution is less than n, and let us consider what happens with an α resolution whose derivation height is n. We have already covered the case where the second resolution is not on the literal obtained from the first resolution.

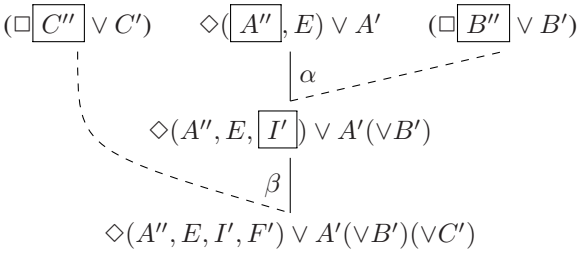
If the literal resolved upon in the second resolution is the one obtained in the first resolution, this means that the first resolution is not a propositional one, nor a $\Box\bot$ one (since in these cases, two literals are discarded and not *replaced* by anything), and thus the second one is not propositional either, since it operates on a modal literal.

If the α-resolution involves a $\Diamond$-literal, say in clause A, then it can be a $\Diamond 1$ or a $\Diamond 2$ resolution on A alone, or a $\Diamond\Box$-resolution with a "side" clause B of the form $B = \Box B'' \vee B'$. Note that we cannot have here $B'' = \bot$, for this would mean that the literals of A and B that are involved would not be replaced by anything, so the literal of A involved in the β-resolution would be another one, a case we have already covered. Thus $B'' \neq \bot$, and the resulting literal is a $\Diamond$-literal containing one new clause I' that is not in the $\Diamond$-literal of A. The resulting $\Diamond$-literal is then resolved upon in the β-resolution, with or without the help of side-clause C, which must then be of the form $C = \Box C'' \vee C'$.

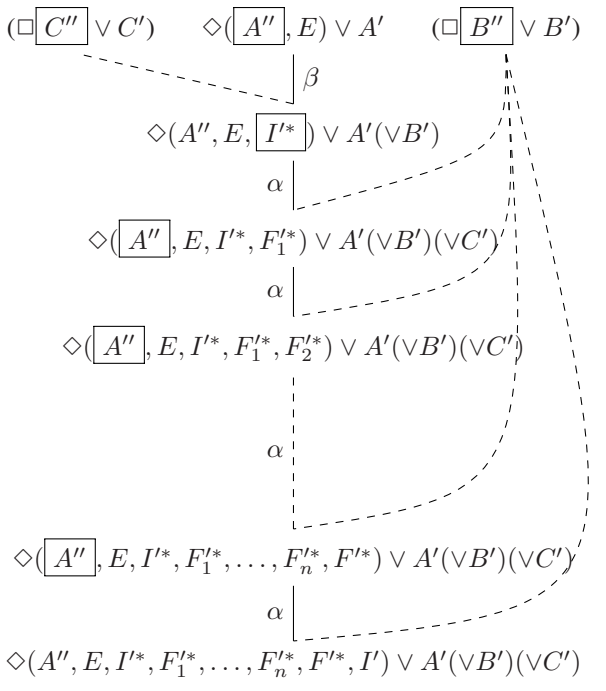
If the β-resolution does not involve I', we can conclude quite easily: in this case, A is of the form $A = \Diamond(E, A'', A''') \vee A'$, where E is a set of clauses and A'' and A''' are the clauses involved in the α- and β-resolution: we have $A''(, B'') \Longrightarrow_\alpha I'$ and $A'''(, C'') \Longrightarrow_\beta F'$, so we also have $\Diamond(E, A'', A''') \vee A'(, \Box C'' \vee C') \Longrightarrow_\beta \Diamond(A'', A''', E, F') \vee A'(\vee C')$ and $\Diamond(A'', A''', E, F') \vee A'(\vee C'), \Box B'' \vee B' \Longrightarrow_\alpha \Diamond(A'', A''', E, I', F') \vee A'(\vee B')(\vee C')$. Pictorially:



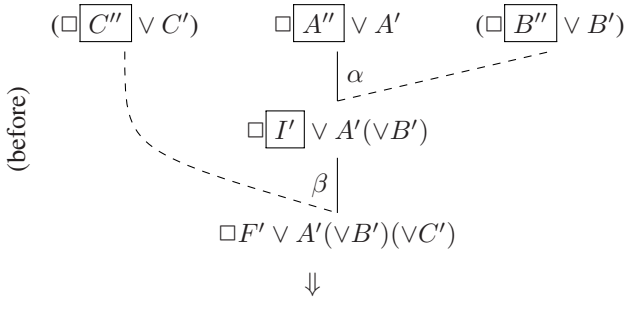
If the β-resolution does involve I', then we'll have to use some inductive argument in order to conclude: A is of the form $A = \Diamond(E, A'') \vee A'$, and we have $A''(, B'') \Longrightarrow_\alpha I'$ and $I'(, C'') \Longrightarrow_\beta F'$:

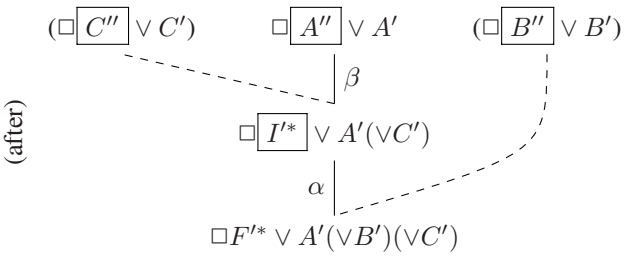


Since the derivation height of $A''(, B'') \Longrightarrow_\alpha I'$ is $n - 1$, by induction hypothesis there exist I'^*, $F_1'^*$, …, $F_n'^*$ (for some $n \geq 0$) and F'^* such that F'^* subsumes F' and $A''(, C'') \Longrightarrow_\beta I'^*$ and $I'^*(, B'') \Longrightarrow_\alpha F_1'^*$ and $F_1'^*(, B'') \Longrightarrow_\alpha F_2'^*$ and … and $F_n'^*(, B'') \Longrightarrow_\alpha F'^*$. Now, we can use another resolution to obtain I', since we still have $A''(, B'') \Longrightarrow_\alpha I'$. Thus we have:

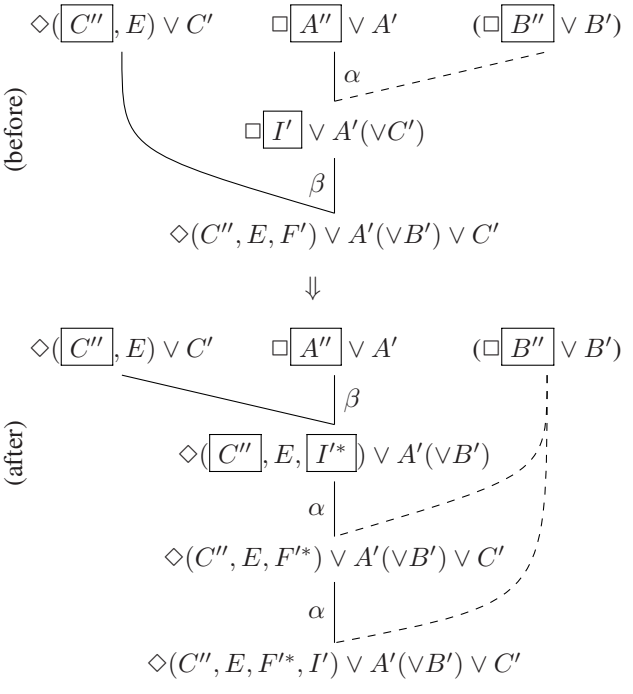


Let us turn now to the case where there is no $\Diamond$-literal involved in the α-resolution: this resolution is derived using a $\Box\Box$-rule or a $\Box$-rule. Then there is a clause A of the form $A = \Box A'' \vee A'$, and possibly a "side"-clause $B = \Box B'' \vee B'$, such that $A''(, B'') \Longrightarrow_\alpha I'$, and thus $I = \Box I' \vee A(\vee B')$. The β-resolution involves the literal produced in the α-resolution (we have studied the other case first). Let us start with the case where this β-resolution does not involve any $\Diamond$-literal: it is a $\Box\Box$- or a $\Box$-resolution, and there may be a clause $C = \Box C'' \vee C'$, with $I'(, C'') \Longrightarrow_\beta F'$, and $F = \Box F' \vee A'(\vee B') \vee (C')$. Using our induction hypothesis again, there are clauses I'^* and F'^* such that $A''(, C'') \Longrightarrow_\beta I'^*$ and $I'^*(, B'') \Longrightarrow_\alpha F'^*$. (For the sake of simplicity, we assume that there are no other intermediate clauses $F_1'^*, \ldots, F_n'^*$; the proof would be very similar with more intermediate clauses.) In this case, we can choose $I^* = \Box I'^* \vee A'(\vee C')$ and $F^* = F'^* \vee A'(\vee B')(\vee C')$:





If the other literal of the β-resolution is a $\Diamond$-literal, we have $C = \Diamond(C'', E) \vee C'$, and $F = \Diamond(C'', E, F') \vee A'(\vee B')(\vee C')$, with $I', C'' \Longrightarrow_\beta F'$. Since $A''(, B'') \Longrightarrow_\alpha I'$, the same inductive hypothesis again allows us to assume that there are clauses I'^* and F'^* such that $A'', C'' \Longrightarrow_\beta I'^*$ and $I'^*(, B'') \Longrightarrow_\alpha F'^*$. In order to get I' in the $\Diamond$ literal after the resolution, we need to perform another α-resolution. In the end, we can choose $I^* = \Diamond(\boxed{C''}, E, \boxed{I'^*}) \vee A'(\vee B')$ and $F^* = \Diamond(C'', E, F'^*, I') \vee A'(\vee B') \vee C'$:



GOAL Agents Instantiate Intention Logic

Koen Hindriks[1] and Wiebe van der Hoek[2]

[1] Delft University of Technology, The Netherlands
k.v.hindriks@tudelft.nl
[2] University of Liverpool, UK
wiebe@csc.liv.ac.uk

Abstract. It is commonly believed there is a big gap between agent logics and computational agent frameworks. In this paper, we show that this gap is not as big as believed by showing that GOAL agents instantiate Intention Logic of Cohen and Levesque. That is, we show that GOAL agent programs can be formally related to Intention Logic. We do so by proving that the GOAL Verification Logic can be embedded into Intention Logic. It follows that (a fragment of) Intention Logic can be used to prove properties of GOAL agents. The work reported is an important step towards the application of standard tools from modal logic for e.g. model checking agent programs. Our results also prove useful for extending the expressiveness of the GOAL agent language. This is illustrated by incorporating temporally extended goals into GOAL agents.

1 Introduction

As has been observed by many others, there is still a considerable gap between logical theories of rational agents and most computational frameworks for such agents [10,12]. Though it is generally hard to connect computational frameworks for rational agents to logics for such agents, in this paper we show that it is possible to formally relate the GOAL agent programming language [4,8] and Intention Logic of Cohen and Levesque [3]. The result proven establishes that GOAL agents instantiate the theory of rational agents as proposed by Intention Logic, although we also argue that the theory needs revision at a number of points.

The motivation behind our work is the observation that there are a number of basic similarities between Intention Logic and the GOAL Verification Logic ("GOAL Logic" for short, see [4]). Most notably, both are based on linear time frames and both incorporate basic notions of a common sense perspective on rational action - beliefs and goals in relation to action. Intention Logic has been proposed as a *theory of the "rational balance" of beliefs, goals, intentions and actions*, inspired by Bratman's theory of intention. It thus proposes a set of rationality principles rational agents should comply with. The GOAL agent programming language is based on and aspires to incorporate similar rationality principles, and has been proposed as a *theory of computation* based on the common-sense notions of belief and goal. Relating both formally thus would be a significant step in bridging the gap between agent theory and engineering.

Establishing a formal connection between GOAL and Intention Logic is useful for a number of reasons. First of all, it connects the GOAL agent programming language

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 232–244, 2008.

to an agent logic in a formally precise sense, contributing to one of the long-standing challenges of agent research of bridging the gap between agent theory and agent programming [10]. It shows that agent logics such as Intention Logic can be applied and used for the verification of properties of computational agents. Conceptually it is interesting to compare the agent concepts and rationality principles incorporated in Intention Logic with those used by GOAL agents. Related to this we show that establishing a formal connection turns out to be useful for extending GOAL agents with temporally extended goals [1]. On top of this, technically, the mapping of GOAL Logic into a standard modal logic is useful since it makes available the rich set of tools available for such logics. These include, for example, tools for model checking, which can be used to achieve one of the main goals of our work - to establish verification tools that can be practically applied to computational rational agents. Finally, combining the frameworks of two approaches also has an effect in the opposite direction: we will argue that assumptions made for Intention Logic can be broadly categorised threefold: those that constitute a basic logic for intention, those that can be conceived of as natural in special situations, and those that seem to be not necessary, or even, not intuitive.

The paper is organized as follows. In Section 2 we briefly introduce the agent programming language GOAL and its verification logic as proposed in [4]. In Section 3 the propositional fragment of Intention Logic used in this paper is introduced. In Section 4 we show that GOAL Logic can be embedded into Intention Logic. In Section 5 we (re)use the embedding proof to show how to incorporate temporally extended goals into GOAL agents. Finally, in Section 6 we conclude the paper and discuss possible directions for future work.

2 The Agent Programming Language GOAL

GOAL agents derive their choice of action from their *beliefs* and their *goals*. GOAL agents consist of four components: (i) a set of beliefs called a *belief base*, (ii) a set of goals called a *goal base*, (iii) a set of action rules, called the *agent program*, and (iv) a set of *action specifications*. The beliefs and goals are drawn from some logical language. The basic ingredients needed are a knowledge representation language and associated inference relation and update operators. Here we follow [4] and throughout the paper we assume a propositional language $\mathcal{L}_0$ (with typical elements ϕ) defined over a set of Atoms with entailment operator $\models$. The beliefs and goals of a GOAL agent define its *mental state*, which needs to satisfy a number of rationality constraints.

Definition 1. (Mental State)
A mental state *of a GOAL agent is a pair* $\langle \Sigma, \Gamma \rangle$ *with* Σ *a belief base and* Γ *a goal base consisting of sentences drawn from a classical propositional language* $\mathcal{L}_0$*, i.e.* $\Sigma, \Gamma \subseteq \mathcal{L}_0$*. A mental state needs to satisfy the following* rationality constraints*:*

- *Belief bases are consistent:* $\Sigma \not\models \mathbf{false}$*,*
- *Individual goals are consistent:* $\forall \gamma \in \Gamma : \gamma \not\models \mathbf{false}$*,*
- *Goals are not believed to be achieved:* $\forall \gamma \in \Gamma : \Sigma \not\models \gamma$*.*

Rational agents are assumed to have consistent beliefs and goals that are not (logically) impossible to achieve which motivates the introduction of the first two rationality constraints. Goals of a GOAL agent are *achievement goals* that the agent wants to

achieve some time in the future. As such, an agent may have multiple achievement goals that taken together are inconsistent but may be achieved in either order over time (cf. [4,7,8]). A GOAL agent is assumed to be committed to achieving these goals. A rational agent however will not invest resources in pursuing goals that are already (completely) achieved, which motivates the third rationality constraint.

In order to be able to decide on its next action a GOAL agent inspects its belief and goal bases. To do so, so-called *mental state conditions* are introduced to reason about the agent's beliefs and goals. The language $\mathcal{L}_m$ of mental state conditions extends $\mathcal{L}_0$ with a modal belief $\mathbf{B}$ and goal $\mathbf{G}$ operator, which can be used to express conditions on the mental state of an agent.

Definition 2. (Mental State Conditions: Syntax)
The language $\mathcal{L}_m$ (with typical elements ψ, ψ') of mental state conditions *is defined by:*

$$\begin{array}{ll} \phi \in \mathcal{L}_0 & ::= \textit{any element in } \mathcal{L}_0 \\ \psi \in \mathcal{L}_m & ::= \mathbf{B}\phi \mid \mathbf{G}\phi \mid \neg\psi \mid \psi \wedge \psi \end{array}$$

The set of mental state conditions consists of Boolean combinations of formulae of the form $\mathbf{B}\phi$ and $\mathbf{G}\phi$ with $\phi \in \mathcal{L}_0$. It is not allowed to nest the operators $\mathbf{B}$ and $\mathbf{G}$ in mental state conditions. Also note that simple propositional formulas without occurrences of $\mathbf{B}$ or $\mathbf{G}$ operators are not mental state conditions. These formulae are called *objective* and are used to represent properties of the agent's environment instead. The semantics of mental state conditions is evaluated with respect to mental states.

Definition 3. (Mental State Conditions: Semantics)
The semantics of mental state conditions is defined relative to a mental state $\langle \Sigma, \Gamma \rangle$.

$$\begin{array}{lll} \langle \Sigma, \Gamma \rangle \models \mathbf{B}\phi & \textit{iff} & \Sigma \models \phi, \\ \langle \Sigma, \Gamma \rangle \models \mathbf{G}\phi & \textit{iff} & \exists \gamma \in \Gamma \textit{ such that } \gamma \models \phi \textit{ and } \Sigma \not\models \phi, \\ \langle \Sigma, \Gamma \rangle \models \neg\psi & \textit{iff} & \langle \Sigma, \Gamma \rangle \not\models \psi, \\ \langle \Sigma, \Gamma \rangle \models \psi \wedge \psi' & \textit{iff} & \langle \Sigma, \Gamma \rangle \models \psi \textit{ and } \langle \Sigma, \Gamma \rangle \models \psi'. \end{array}$$

The semantics of the goal operator $\mathbf{G}$ defines an agent's achievement goals as those propositions that follow from a single goal in the agent's goal base that is not believed to be the case; in other words, $\mathbf{G}\phi$ expresses that ϕ is an achievement goal in this sense.

GOAL agents select actions using a rule-based action selection mechanism. In the remainder, we assume a set of actions A (with typical element α, α') has been provided. *Action rules* of the form $\textbf{if } \psi \textbf{ then } \alpha$ are used to specify that action α can be performed, or, is *enabled*, whenever condition ψ holds, where ψ is a mental state condition. This mechanism allows agents to derive their choice of action from their beliefs and goals. The semantics of action selection and execution are formally specified in GOAL by means of an operational semantics; here, however, we abstract from the formal details (see [4]) and we will represent action selection implicitly by means of action occurrences in a set of possible traces. A *trace* simply is a sequence of mental states and actions.

Definition 4. (Trace)
A trace *t is an infinite sequence $m_0, \alpha_0, m_1, \alpha_1, \ldots$ of mental states m_i and actions α_i. We also write t_i^m to denote the ith mental state and t_i^a to denote the ith action.*

Intuitively, a trace corresponding to a possible computation of a GOAL agent needs to start with a mental state that corresponds to the initial state of the GOAL agent. The changes in mental states over time are the result of executing actions (which ideally correspond to changes in the agent's environment). Action rules and preconditions do not need to determine a unique action to be taken by the agent at a time point. The semantics associated with the action selection and execution of a GOAL agent thus does not define a unique computation but corresponds to a set of computations. This motivates defining the meaning of a GOAL agent $\mathcal{A}$ as a set of traces, in line with the fact that we abstract from the semantics of action selection and execution in this paper.

2.1 GOAL Logic

To obtain a verification logic for GOAL agents temporal operators are added on top of mental state conditions to be able to express temporal properties over traces. Additionally an operator $\mathbf{start}$ is introduced to be able to pinpoint the start of a trace.

Definition 5. (Temporal Language: Syntax)
The temporal language $\mathcal{L}_G$ *(with typical elements* χ, χ'*) is defined by:*

$$\chi \in \mathcal{L}_G ::= \mathbf{start} \mid \psi \in \mathcal{L}_m \mid \neg\chi \mid \chi \wedge \chi \mid \chi \ \mathbf{until}\ \chi \mid [\alpha \in A]\chi$$

The semantics of $\mathcal{L}_G$ is defined relative to an agent $\mathcal{A}$, trace $t \in \mathcal{A}$ and time point i.

Definition 6. (Temporal Language: Semantics)
The truth conditions of sentences from $\mathcal{L}_G$ *given a GOAL agent* $\mathcal{A}$*, trace* $t \in \mathcal{A}$ *and time point* i *are inductively defined by:*

$$\begin{array}{ll}
\mathcal{A}, t, i \models \mathbf{start} & \textit{iff } i = 0, \\
\mathcal{A}, t, i \models \mathbf{B}\phi & \textit{iff } t_i^m \models \mathbf{B}\phi, \\
\mathcal{A}, t, i \models \mathbf{G}\phi & \textit{iff } t_i^m \models \mathbf{G}\phi, \\
\mathcal{A}, t, i \models \neg\varphi & \textit{iff } \mathcal{A}, t, i \not\models \varphi, \\
\mathcal{A}, t, i \models \varphi \wedge \psi & \textit{iff } \mathcal{A}, t, i \models \varphi \textit{ and } \mathcal{A}, t, i \models \psi, \\
\mathcal{A}, t, i \models \varphi \ \mathbf{until}\ \psi & \textit{iff } \exists j \geq i : \mathcal{A}, t, j \models \psi \textit{ and } \forall i \leq k < j : \mathcal{A}, t, k \models \varphi, \\
\mathcal{A}, t, i \models [\alpha]\varphi & \textit{iff } \forall t \in \mathcal{A}(t_i^a = \alpha \Rightarrow \mathcal{A}, t, i+1 \models \varphi).
\end{array}$$

Note that formulas of the form $[\alpha]\varphi$ specify *universal action postconditions*, in particular, we have $\mathcal{A}, t, i \models [\alpha]\varphi$ iff $\mathcal{A}, t', i' \models [\alpha]\varphi$ iff $\mathcal{A} \models [\alpha]\varphi$. This operator allows to define the Hoare system for GOAL which was proven complete in [4] and facilitates reasoning about actions. This operator is crucial in GOAL Logic to be able to compositionally prove properties of all traces induced by a GOAL agent [4].

3 Basic Intention Logic

Our interest in this paper is in the *single-agent, propositional fragment* of Intention Logic without dynamic (composition) operators such as sequential composition. In essence, Intention Logic can be considered a single-agent logic (cf. [12]) and the single agent restriction boils down to excluding multiple agent labels and variables ranging

over such labels from the logical language. The restriction to the propositional fragment implies that we do not introduce quantifiers and variables ranging over events, agents or domains. Temporal operators are also introduced explicitly in the language rather than defining these as rather complex quantifications over events. The fragment of Intention Logic introduced here is referred to henceforth as *Basic Intention Logic*, or sometimes also simply as Intention Logic.

Definition 7. (Basic Intention Logic: Syntax)
The language $\mathcal{L}_{BI}$ is defined by:

$$\begin{array}{l}
\alpha ::= \textit{any element from } A \mid \mathsf{IF}\ \varphi\ \mathsf{THEN}\ \alpha\ \mathsf{ELSE\ NIL},\\
\phi ::= \textit{any element from } \mathsf{Atom},\\
\varphi ::= \phi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{BEL}\ \varphi \mid \mathsf{GOAL}\ \varphi \mid \mathsf{HAPPENS}\ \alpha \mid\\
\qquad \mathsf{DONE}\ \alpha \mid \mathbf{t} \mid \mathsf{BEFORE}\ \varphi\ \varphi \mid E\varphi,\\
\mathbf{t} ::= \textit{any non-negative numeral } (\mathbf{0}, \mathbf{1}, \ldots)
\end{array}$$

The main modification made to Intention Logic is the addition of a global modal operator E (cf. [2]). The operator HAPPENS is too weak to reason about *all possible effects* of executing an action which is crucial for verifying properties of the behaviour of an agent program (compare the dynamic operator $[\alpha]\chi$ introduced above and the usual dynamic modality in Dynamic Logic [6]). The standard abbreviations are used for true and disjunction $\vee$. Some additional abbreviations used are:

$$\mathsf{UNTIL}\ \varphi\ \psi \stackrel{df}{=} \neg(\mathsf{BEFORE}\ \psi\ \neg\varphi), \qquad \Diamond\varphi \stackrel{df}{=} (\mathsf{true\ UNTIL}\ \varphi), \qquad \Box\varphi \stackrel{df}{=} \neg\Diamond\neg\varphi$$
$$\mathsf{KNOW}\ \varphi \stackrel{df}{=} \varphi \wedge \mathsf{BEL}\ \varphi, \qquad \mathsf{KNOWIF}\ \varphi \stackrel{df}{=} \mathsf{KNOW}\ \varphi \vee \mathsf{KNOW}\ \neg\varphi.$$

After introducing the fragment we refer to as *Basic Intention Logic*, the question remains how much of the *theory* of Intention Logic about rational agency survives. As it will turn out, a large part can be (re)formulated by using temporal operators only. This issue will be revisited at the end of this Section.

3.1 A Run-Based Semantics for Intention Logic

Semantically we first introduce a run-based semantics for Intention Logic and then discuss how our semantics relates to that introduced in [3]. Different from [7] we use standard linear orders $\mathbb{L}$ to define models for Intention Logic to ensure our models have the same basic structure as traces of GOAL agents. Here, we will restrict ourselves to $\mathbb{L} = \langle \mathbb{N}, < \rangle$ and $\mathbb{L} = \langle \mathbb{Z}, < \rangle$. We use linear orders to define the concept of a *run*.

Definition 8. (Run-Based Model)
Let an arbitrary set of labels S also called states *be given. A* run based on S and A *is a function $r : \mathbb{L} \to (S \times A)$ that assigns to every time point a state-action pair. Given $n \in \mathbb{L}$, we will write r_n^{st} for the first component of $r(n)$, and r_n^{ac} for the second. The set of runs based on S and A is denoted $\mathcal{R}(S, A)$.*

A run-based model M *(over* Atoms*) is a tuple $M = \langle S, \mathbb{L}, B, G, V \rangle$, where*

- *S is a non-empty set of* states*;*
- *$\mathbb{L}$ is a linear order;*

- $B \subseteq \mathcal{R} \times \mathbb{L} \times \mathcal{R} \times \mathbb{L}$ *is a Euclidean, transitive and serial belief accessibility relation,*
- $G \subseteq \mathcal{R} \times \mathbb{L} \times \mathcal{R} \times \mathbb{L}$ *is a serial goal accessibility relation, and*
- $V : S \rightarrow \mathsf{Atoms}$.

The semantics of Basic Intention Logic can now be defined using run-based models.

Definition 9. (Run-Based Semantics for Basic Intention Logic)
Let $M = \langle S, \mathbb{L}, B, G, V \rangle$ be a run-based model, $r \in \mathcal{R}$, and $n \in \mathbb{L}$. Then the satisfaction relation $\models$ relative to M is defined by:

$$
\begin{array}{ll}
M, r, n \models p & \textit{iff } p \in V(r^{st}_n) \\
M, r, n \models \neg\varphi & \textit{iff } M, r, n \not\models \varphi \\
M, r, n \models \varphi \wedge \varphi' & \textit{iff } M, r, n \models \varphi \textit{ and } M, r, n \models \varphi' \\
M, r, n \models \mathbf{t} & \textit{iff } \mathbf{t} \textit{ denotes } n, \\
M, r, n \models \mathsf{DONE}\ \alpha & \textit{iff } \exists j \in \mathbb{L}\ M, r, j[\![\alpha]\!]n \\
M, r, n \models \mathsf{HAPPENS}\ \alpha & \textit{iff } \exists j \in \mathbb{L}\ M, r, n[\![\alpha]\!]j \\
M, r, n \models \mathsf{BEL}\ \varphi & \textit{iff } \forall r', n'(B(r, n, r', n') \Rightarrow M, r', n' \models \varphi) \\
M, r, n \models \mathsf{GOAL}\ \varphi & \textit{iff } \forall r' n'(G(r, n, r', n') \Rightarrow M, r', n' \models \varphi) \\
M, r, n \models \mathsf{BEFORE}\ \varphi\ \psi & \textit{iff } \forall j \geq n(M, r, j \models \psi \Rightarrow \exists i \leq j(M, r, i \models \varphi)) \\
M, r, n \models E\varphi & \textit{iff } \exists r', n'\ \ M, r', n' \models \varphi
\end{array}
$$

where $M, r, n[\![\alpha]\!]n'$, to interpret DONE α *and* HAPPENS α*, is defined as follows:*

1. $M, r, n[\![\alpha]\!]n'$ *iff* $r^{ac}_n = \alpha$ *and* $n' = n + 1$.
2. $M, r, n[\![\mathsf{IF}\ \varphi\ \mathsf{THEN}\ \alpha\ \mathsf{ELSE\ NIL}]\!]n'$ *iff* $M, r, n \models \varphi \Rightarrow M, r, n[\![\alpha]\!]n'$.

Note in particular the definition of semantics of the global modality E, which is an extension of Intention Logic: this operator allows inspection of arbitrary states within a model, which is useful to translate the dynamic operator $[\alpha]\chi$ of GOAL Logic into Intention Logic.

3.2 CL Models for Intention Logic

How do our Run-Based Models (RBM, from now) compare to the Cohen & Levesque Models, as presented in [3] (CLM, henceforth)?

Observation 1. *The following relates* RBM *with* CLM*:*

1. CLM *models are a special case of* RBM *models in the following sense: In* CLM *models,*
   (a) $\mathbb{L}$ *is taken to be* $\mathbb{Z}$
   (b) agents know the correct time: *If* $B(r, n, r', n')$ *then* $n = n'$
   (c) agents "want" the current time: *If* $G(r, n, r', n')$ *then* $n = n'$
   (d) G *and* B *are related through* realism*:* $G \subseteq B$
   (e) a run is of type $\mathbb{L} \rightarrow A$
   (f) runs are determined by their action part, i.e.,

$$\forall r, r'(\forall n : r^{ac}_n = r'^{ac}_n \Rightarrow \forall n\ V(r^{st}_n) = V(r'^{st}_n)$$

(g) *agents remember the last atomic action they have done: if* $B(r,n,r',n')$ *then* $n = n'$ *and* $r_n^{ac} = r_n'^{ac}$.
(h) *assume the property of* No persistence / deferral forever, *see below.*

2. *However,* RBM *models are also a specialisation of* CLM *models:*
(a) CLM *allows for quantification over a domain of objects and events;*
(b) CLM *models have a richer notion of composed actions, and accordingly an extended definition of* $M, r, n[\![\alpha]\!]n'$.
(c) CLM *models are defined for multiple agents.*

Some of the differences mentioned above are merely a matter of choice or design. For instance, it is straightforward to extend the notion of Run-Based Model in such a way that they encompass item 2(b,c) of Observation 1. As regarding item 1, there are some deeper issues involved. As to 1a, it seems natural for computational systems to assume that computations have a start somewhere. Syntactically, item 1a amounts to the requirement that there is always some atomic action α for which $\mathsf{DONE}\ \alpha$ holds.[1] To assume that agents know the correct time (1b) makes sense in many scenario's, and, given that an agent knows the time, it does not make sense to have a "goal" that the time were different. Where realism of CLM ensures $\mathsf{BEL}\ \varphi \rightarrow \mathsf{GOAL}\ \varphi$, the weak realism of RBM amounts to $\mathsf{BEL}\ \varphi \rightarrow \neg\mathsf{GOAL}\ \neg\varphi$. We don't think realism is a very realistic(!) assumption, and we even think that Cohen and Levesque had *weak realism* in mind when they presented their semantics ([3][p. 227]):

> *. . . 'the worlds that are consistent with what the agent has chosen are not ruled out by his beliefs. Without this constraint, the agent could choose world involving (for example) future events that he believes will never happen.'*

Hence, we will assume that R and G satisfy *weak realism*: for every $r \in \mathcal{R}$, and $n \in \mathbb{L}$, there is a $r' \in \mathcal{R}$ and $n' \in \mathbb{L}$ such that $(r,n,r',n') \in G \cap B$.

Let us now consider item 1f, which is related to item 1e which restricts runs to $\mathbb{L} \rightarrow A$. Suppose we take runs as basic entities, like in CLM. This does not do justice to the intensional notion of the logic, as can be seen as follows. Suppose that we have only one atom $p \in \mathsf{Atoms}$, and two basic actions α, α'. Let $\widehat{\mathsf{BEL}}\ \varphi$ be $\neg\mathsf{BEL}\ \neg\varphi$: the agent considers φ doxastically possible. Let ψ be $\mathbf{0} \wedge \Box(p \wedge \mathsf{DONE}\ \alpha)$. Now consider $\widehat{\mathsf{BEL}}\ (\psi \wedge \mathsf{GOAL}\ p) \wedge \widehat{\mathsf{BEL}}\ (\psi \wedge \neg\mathsf{GOAL}\ p)$. This is not satisfiable in CLM, since ψ determines a unique run, and what the goals and beliefs of an agent are is determined by the run. More natuaral examples present themselves in the multi-agent case, where we would have for instance that $\widehat{\mathsf{BEL}}_1(\psi \wedge \mathsf{BEL}_2 p) \wedge \widehat{\mathsf{BEL}}_1(\psi \wedge \neg\mathsf{BEL}_2 p)$ is unsatisfiable[2].

Given that CLM models identify runs and paths, and a run in CLM is of type $\mathbb{L} \rightarrow A$, already brings a problem to the fore that is more basic than on the intensional level. In CLM, a valuation Φ checks whether $\Phi(p,\sigma,n)$ holds, where p is an atomic proposition, σ is a 'event-run': $\mathbb{Z} \rightarrow A$ and $n \in \mathbb{Z}$. But this implies that the truth of atomic propositions

[1] Since we do not have quantification over events in the propositional version of IL, we assume that all transitions are labeled with an atomic action.

[2] for readers familiar with modal epistemic logic, this is exactly the reason why states are not identified with valuations: there would not be enough valuations (in case of one atom) to satisfy $\neg K_1 \neg(p \wedge K_2 p) \wedge \neg K_1 \neg(p \wedge \neg K_2 p)$

(and hence, of objective formulas) is compeletely determined once we know which actions are taken along σ. In other words, it is not possible to have two event-runs that agree on all the actions, but still objective formulas along them differ. Suppose the event α represents the throwing of a dice, ($\sigma(n) = \alpha$, for all n) and that $p_i (i \leq 6)$ represents the outcome. Now, let Φ determine how the propositions p_i are distributed over σ, say $\Phi(p_i, \sigma, n)$ iff $i = n \bmod 6$. Now, the type of Φ dictates that there *cannot* be another event run σ' in which a dice is continuously thrown *but the outcomes are different!* In particular, this implies that if our agent knows that α always happened and will always happen, he will also know all the outcomes (there is no alternative run with the same actions and different outcomes). Summarising (let G^{-1} refer to the past):

$$\mathsf{KNOW}\ (\Box\ \mathsf{HAPPENS}\ \alpha \wedge \mathsf{G}^{-1}\mathsf{DONE}\ \alpha) \rightarrow \mathsf{KNOWIF}\ \varphi \tag{1}$$

Property 1g of Observation 1 is implicitly imposed by [3] since they require

$$[3, \textbf{Assumption 3.20}]\ \models (\mathsf{DONE}\ \alpha) \leftrightarrow (\mathsf{BEL}\ (\mathsf{DONE}\ \alpha))$$

Let us now look at 1h. This is the semantic counterpart of another assumption made in [3], motivated by the fact that an agent should not endlessly pursue the same goal:

$$[3, \textbf{Assumption 3.25}]\ \models \Diamond\neg(\mathsf{GOAL}\ (\neg\varphi \wedge \Diamond\varphi))$$

Writing $\widehat{\mathsf{GOAL}}\ \varphi$ for $\neg\mathsf{GOAL}\ \neg\varphi$, it is not hard to see that this is equivalent to:

$$\models \Diamond\ \widehat{\mathsf{GOAL}}\ (\varphi \rightarrow \Box\varphi) \tag{2}$$

However, (2) *as a scheme* corresponds, in the sense of modal logic [2] to a semantic property that is incompatible with the models we are currently looking at. Note that for $\varphi \rightarrow \Box\varphi$ to be true in a world x corresponds to the fact that $\forall y(x \leq y \rightarrow y = x)$ (there is at most one instance that is later than x, and this is x itself. Then for $\Diamond\ \widehat{\mathsf{GOAL}}\ (\varphi \rightarrow \Box\varphi)$ to be true in all worlds z corresponds to

$$\forall z \exists u \exists x(z \leq u\ \&\ (Gux\ \&\ \forall y(x \leq y \rightarrow y = x))) \tag{3}$$

In words: for every time point, there is a future time point with a GOAL-accessible point, such that the latter point only has itself as a future successor. This property is incompatible with our models (and indeed, with CLM models), since (1) time is supposed to go on forever, and (2) we have 'nominals' that are true at only one time point: in the x state above, some time expression $\mathbf{x}$ must be only in x itself, and not its successors.

[3, **Assumption 3.25**] expresses that 'there is a future point such that in some goal-accessible world, no goal is true anymore', while the intuition [3] seem to want to capture is 'for every goal there will be a time point in the future that it is dropped'. The latter seems hard to be conceived of as a structural property on models, and indeed, we think it should be a property of the protocol, or behaviour, of the agent.

Summarising, for our semantics, we assume time has a starting point, and that agents know and want the time. The other restrictions 1d - 1h are either properties that give undesired properties (1d, 1e, 1f, 1h), or can be added on top of a basic class of models (1g).

Definition 10 (Run-Based IL-Models). *The class of Run-Based Basic Intention Logic Models,* RBBILM *for short, is the class of Run-Based Models* $M = \langle S, \mathbb{L}, B, G, V \rangle$ *such that:*

1. $\mathbb{L} = \mathbb{Z}$
2. *agents know the correct time*
3. *agents want the correct time*
4. B *and* G *are connected through weak realism.*

Validity in the class RBBILM *is denoted* $\models_{BI}$.

4 Connecting GOAL and Intention Logic

In this Section we show how to formally relate GOAL and Intention Logic. First, we define a translation function from GOAL into Intention Logic. Except for the goal operator and the dynamic modality of GOAL Logic this is straightforward. The main result we want to prove is that properties proven to hold in one logic are preserved under translation from that logic to the other. We do so by showing that satisfaction of a formula is preserved under translation.

Definition 11. (Translating $\mathcal{L}_G$ into $\mathcal{L}_{BI}$)
The translation function τ *mapping GOAL Logic formulae and action rules onto Intention Logic formulae is defined by:*

$$
\begin{array}{ll}
\tau(\mathbf{start}) & = \mathbf{0}, \\
\tau(\mathbf{B}\phi) & = \mathsf{BEL}\ \phi, \\
\tau(\mathbf{G}\phi) & = \mathsf{GOAL}\ \Diamond\phi \wedge \neg \mathsf{BEL}\ \phi, \\
\tau(\neg\chi) & = \neg\tau(\chi), \\
\tau(\chi_1 \wedge \chi_2) & = \tau(\chi_1) \wedge \tau(\chi_2), \\
\tau(\chi_1\ \mathbf{until}\ \chi_2) & = \tau(\chi_1) \mathsf{UNTIL}\ \tau(\chi_2), \\
\tau([\alpha]\chi) & = U(\mathsf{DONE}\ \alpha \rightarrow \tau(\chi)), \\
\tau(\mathbf{if}\ \psi\ \mathbf{then}\ \alpha) & = \mathsf{IF}\ \tau(\psi)\ \mathsf{THEN}\ \alpha\ \mathsf{ELSE\ NIL}.
\end{array}
$$

The most interesting case in the definition of the translation function τ is the translation of $\mathbf{G}\phi$. An achievement goal in GOAL requires that the agent does not believe ϕ to be the case, whereas [3] require the agent to believe that ϕ is not the case. Whereas the goal operator $\mathbf{G}$ does not satsify axiom D (cf. [4]; see also [7] for a discussion), the achievement goal operator of [3] does, implying that an agent cannot have inconsistent achievement goals.

The proof showing that satisfaction is preserved under translation is based on model constructions. Lemma 1 shows how to derive a GOAL Logic model (a trace) from an RBBILM model that preserves satisfaction of formulae from GOAL Logic, whereas Lemma 2 shows how to construct an RBBILM model from a GOAL trace. Theorem 2 states our main result that satisfaction is preserved under translation, which shows that Basic Intention Logic can be used to prove properties of GOAL agents.

Lemma 1. *Let $M = \langle S, \mathbb{L}, B, G, V\rangle$ be an* RBBILM *model. Then there is a GOAL agent $\mathcal{A}$ and a function f from runs to traces such that the set of traces in $\mathcal{A}$ is $\{f(r) \mid r \in \mathcal{R}(S, A)\}$ and for all $\varphi \in \mathcal{L}_G$:*

$$M, r, n \models_{BI} \tau(\varphi) \textit{ iff } \mathcal{A}, f(r), n \models_G \varphi$$

Proof. We need to construct a GOAL trace $f(r) = t = m_0, \alpha_0, m_1, \alpha_1, \ldots$ for every run $r \in \mathcal{R}$, where each mental state m_i is of the form $\langle \Sigma_i, \Gamma_i\rangle$. The components can be derived from r as follows:

- $\Sigma_i = \{\phi \in \mathcal{L}_0 \mid M, r, i \models_{BI} \mathsf{BEL}\ \phi\}$,
- $\Gamma_i = \{\phi \in \mathcal{L}_0 \mid M, r, i \models_{BI} \mathsf{GOAL}\ \Diamond\phi \wedge \neg\mathsf{BEL}\ \phi\}$, and
- $\alpha_i = r_i^{ac}$.

Since the relation B in RBBILM models is serial, each Σ_i is consistent. For a similar reason every $\gamma \in \Gamma_i$ is consistent. Moreover, by constuction of Γ_i, we have $\forall\gamma \in \Gamma_i, \Sigma \not\models \gamma$. We now show the equivalence of $\tau(\varphi)$ in M, r, n with that of $f(r), n$ in $\mathcal{A}$ by induction on φ as follows.

If φ is $\mathbf{start}$, we have $M, r, n \models_{BI} \mathbf{0}$ iff $n = 0$ iff $\mathcal{A}, f(r), 0 \models_G \mathbf{start}$. For the intensional operators, the equivalence follows immediately from the definition of mental states in the trace $f(r)$. Finally, let $\varphi = [\alpha]\chi$. Then $M, r, n \models_{BI} U((\mathsf{DONE}\ \alpha) \rightarrow \tau(\chi))$ iff for every run r' and n', we have $M, r', n' \models_{BI} \mathsf{DONE}\ \alpha \rightarrow \tau(\chi)$. Now let t' be an arbitrary trace in $\mathcal{A}$, and suppose $t_i'^a = \alpha$. Obviously, this trace must be the image of a run r' for which $r_i'^{ac} = \alpha_i$. But then, $M, r', i+1 \models_{BI} \mathsf{DONE}\ \alpha$ and, hence $M, r', i+1 \models_{BI} \tau(\chi)$. By induction, $\mathcal{A}, t', i+1 \models_G \chi$. This demonstrates $\mathcal{A}, t, n \models_G [\alpha]\chi$. The other direction is similar. □

Lemma 2. *Let $\mathcal{A}$ be an agent, that is, a set of traces. Then we can construct an* RBBILM *model $M = \langle S, \mathbb{L}, B, G, V\rangle$ such that there is a function $g : \mathcal{A} \rightarrow \mathcal{R}$ satisfying, for every $\varphi \in \mathcal{L}_G$ and every $n \in \mathbb{N}$:*

$$\mathcal{A}, t, n \models_G \varphi \textit{ iff } M, g(t), n \models_{BI} \tau(\varphi)$$

Proof. Let $Constraints = \{[\alpha]\chi \mid \mathcal{A} \models_G [\alpha]\chi\}$. Let ε be an action symbol not occurring in $\mathcal{L}_G$. Call a run r *minimal* if for all n, $V(r_n^{st}) = \emptyset$ and $r_n^{ac} = \varepsilon$. Call a run r *peak-once* if it is like a minimal run, except that for at most one $k \in \mathbb{N}$, we can have $V(r_k^{st}) \neq \emptyset$. Given a trace t, we have to find its associated run $g(t)$. Let $t = m_0, \alpha_0, m_1, \alpha_1, \ldots$. For the run $g(t)$, we put $g(t)_i^{ac} = \alpha_i$. Let the mental state at t_i be $\langle \Sigma_i, \Gamma_i\rangle$. For every valuation π for which $\pi \models \Sigma_i$, add a state $\langle t, i, \pi\rangle$. Put $V(\langle t, i, \pi\rangle) = \pi$ and, for every such state $\langle t, i, \pi\rangle$, add a peak-once run r' such that $r_i'^{st} = \langle t, i, \pi\rangle$. Put $B(g(t), i)(r', i)$ for each such run. This procedure guarantees that

$$\text{For all } \phi \;\; M, g(t), i \models_{BI} \mathsf{BEL}\ \phi \text{ iff } \phi \in \Sigma_i \tag{4}$$

For the goals Γ_i, we distinguish two cases. First, suppose $\Gamma_i = \emptyset$. Then, for the goal-associated runs in $g(t), i$ we take exactly the belief-associated runs as described above. Apart from weak realism, this guarantees

$$\Gamma_i = \emptyset \Leftrightarrow \text{ for no } \phi : M, g(t), i \models_{BI} \mathsf{GOAL}\ \Diamond\phi \wedge \neg\mathsf{BEL}\ \phi \tag{5}$$

Now, if $\Gamma_i \neq \emptyset$, let $\gamma_1, \gamma_2, \ldots$ be an infinite enumeration of all elementes of Γ_i: if Γ_i only has a finite number h of elements, we put $\gamma_{h+j} = \gamma_j$. Since each γ_j is consistent, it comes with a set of propositional valuations Π_j. Let k be the biggest cardinality of those sets Π_j, which could be an element of $\mathbb{N}$ or else ∞. Now we associate k goal-accessible runs r with $g(t), i$ such that for every $m, m' > i$, $V(r^{st}_m), V(r^{st}_{m'})$ are valuations from Π_m, whenever $\gamma_m = \gamma_{m'}$ then $V(r^{st}_m) = V(r^{st}_{m'})$, and, conversely, every valuation in Π_m occurs in at least one goal-accessible run. Since the language $\mathcal{L}_G$ cannot talk about the past, it does not matter how such a run looks like at $j \leq i$, although, in order to obtain weak realism, we take care that there is at least one of the goal-runs r_g just created for which $r_g(i) = r_b(i)$, where r_b is one of the belief-accessible runs. We finally specify $r^{ac}_n = \varepsilon$ for all n, for all such runs r. Since we know that $\langle \Sigma_i, \Gamma_i \rangle \models \mathbf{G}\phi$ implies that $\phi \notin \Sigma_i$, this procedure guarantees that

$$\text{For all } \phi \; \phi \in \Gamma_i \text{ iff } M, g(t), i \models_{BI} \mathsf{GOAL}\, \Diamond\phi \wedge \neg \mathsf{BEL}\, \phi \tag{6}$$

Now, the model M is built by taking all runs $g(t)$ from $t \in \mathcal{A}$, and adding the associated goal and belief runs (the states that we need are defined when we defined the runs).

The proof of the overall claim again follows using induction on φ, where the intensional operators follow directly from (4), (5) and (6). The only interesting remaining case are $Constraints$. So let us consider $\varphi = [\alpha]\chi$, and the property proven for χ. Suppose furthermore $\mathcal{A}, t, n \models_G [\alpha]\chi$. This means that for all t' and m that $t'^a_m = \alpha \Rightarrow \mathcal{A}, t', m+1 \models_G \chi$. In M, the only runs r for which there is an i such that $M, r, i+1 \models_{BI} \mathsf{DONE}\, \alpha$ holds, are runs for which there is a trace t such that $r = g(t)$ and in t, α_i equals α (since the constructed goal and belief runs only refer to action ε). But using induction, we have $M, g(t), m+1 \models_{BI} \tau(\chi)$, which completes the proof. □

Theorem 2. *GOAL semantics $\models_G$ and semantics of Run-Based Basic Intention Logic $\models_{BI}$ are equivalent for the $\mathcal{L}_G$ and $\tau(\mathcal{L}_G)$.*

Proof. Imediate from Lemma 1 and 2. □

5 Extending GOAL Agents with Temporally Extended Goals

The mapping of goals in the GOAL language onto Intention Logic as in Definition 11 shows that these are naturally interpreted as *achievement goals*, as originally intended [4,8]. The future-directed interpretation of such goals is left implicit in GOAL whereas it is made explicit in the definition of such goals in Intention Logic. By making the temporal component explicit it is straightforward to define other goal types in Intention Logic. For example, *maintenance goals* can be defined as $\mathsf{GOAL}\ (\Box\phi)$. The idea to introduce a primitive "goal" operator GOAL (or $Choice$ as [7] call it) in Intention Logic that allows defining various goal types can be introduced in GOAL as well to increase expressivity [9]. In this Section we show how we can apply the result of the previous Section to extend GOAL with *temporally extended goals* [1] while still maintaining the connection between GOAL Logic and Intention Logic.

To this end, we now allow *pure* temporal formulae φ in the belief and goal base of GOAL agents. As the idea is to define achievement and other goals now in GOAL in

the same way as in Intention Logic, the semantics of the goal operator $\mathbf{G}$ in GOAL is modified analogously, and is now simply defined as:

$$\langle \Sigma, \Gamma \rangle \models \mathbf{B}\phi \text{ iff } \Sigma \models_{LTL} \phi,$$
$$\langle \Sigma, \Gamma \rangle \models \mathbf{G}\phi \text{ iff } \Gamma \models_{LTL} \phi.$$

As we now allow temporal formulae ϕ without occurrences of other modal operators in both the belief and goal base, the entailment relation of linear temporal logic is used [5]. It is clear that with these operators we can reintroduce the notion of an achievement goal by definition as $\mathbf{G}\Diamond\phi \wedge \neg\mathbf{B}\phi$. Moreover we no longer require as in Definition 1 that individual goals in a goal base Γ are consistent (this is now taken care of by the temporal operators) but instead require that Γ itself is consistent. A further simplification as a result of this modified setup is that the rationality constraint of Definition 1 that goals are not believed to be the case is no longer needed as this now follows by definition.[3]

It turns out that to show that the connection with Intention Logic is maintained requires only minor modifications of the proofs provided in Section 4 and actually simplifies matters somewhat. The proof of Lemma 1 only requires a modification of the derivation of mental states from a run r, as follows:

- $\Sigma_i = \{\phi \in \mathcal{L}_{LTL} \mid M, r, i \models_{BI} \mathsf{BEL}\ \phi\}$,
- $\Gamma_i = \{\phi \in \mathcal{L}_{LTL} \mid M, r, i \models_{BI} \mathsf{GOAL}\ \phi\}$.

As for Lemma 2, since in the new setup (see definitions above) belief and goal bases have the same logical properties there is no need anymore to distinguish them in the proof. It thus suffices to show how to construct a run r such that we have (4^*) $M, g(t), i \models_{BI} \mathsf{BEL}\ \phi$ iff $\Sigma_i \models_{LTL} \phi$ (cf. Lemma 2). As before, for a given trace t we have to find an associated run $g(t)$. Call a run r *silent* if it consists of ϵ-steps only, i.e. $r_n^{ac} = \epsilon$ for all n. Then put $B(g(t), i, r, i)$ for each silent run such that $M, r, i \models \Sigma_i$. This procedure guarantees (4^*). The same procedure can be used to prove (5^*) $M, g(t), i \models_{BI} \mathsf{GOAL}\ \phi$ iff $\Gamma_i \models_{LTL} \phi$, and we are done. Finally, by changing the translation mapping of Definition 11 for $\mathbf{G}\phi$ to $\mathsf{GOAL}\ \phi$ we obtain:

Theorem 3. *The GOAL semantics $\models_G$ and semantics of Run-Based Basic Intention Logic $\models_{BI}$ are equivalent for the languages $\mathcal{L}_G^{LTL}$ and $\tau(\mathcal{L}_G^{LTL})$ that include temporally extended goals and beliefs.*

6 Conclusion

We showed that GOAL agents instantiate Intention Logic and can be formally related by means of translating GOAL Logic into Intention Logic. Two important results follow: (i) GOAL Logic is equivalent to a propositional fragment of Intention Logic, and (ii)

[3] There remains however the problem of how and when to remove goals from the goal base of an agent. In [9] a progression operator has been introduced as a solution to this problem (see also [1]). In the setup of this Section the main difference between the belief and goal base is this automatic mechanism of removing goals from the goal base, which represents the default *commitment strategy* of an agent (cf. [4, 8, 11]).

this fragment - a standard normal tense logic - can be used to *verify* GOAL agents using a Hoare logic for actions performed by GOAL agents (using additional derivation rules for verification introduced in [4]). The result proved useful for incorporating temporally extended goals into GOAL while maintaining the connection with Intention Logic.

We argued that Intention Logic at a number of points needs revision. In particular, we argued that the principle of No Persistence Forever that requires an agent to drop every one of its goals sometime is too strong. Moreover, the notion of achievement goals used in GOAL is slightly different from that of [3] and more in line with that proposed in [7].

Future work will involve applying our results in model checking of GOAL agents. Conceptually we are interested in including preferences into the language while maintaining a logical connection with a standard modal logic, which involves extensions to the programming language GOAL [9] as well as to Basic Intention Logic. The additional expressivity introduced by incorporating temporally extended goals and temporal formulae into the belief base of GOAL agents also raises many new questions about goal persistence and the operationalization of, for example, maintenance goals [9].

References

1. Bacchus, F., Kabanza, F.: Planning for temporally extended goals. Annals of Mathematics and Artificial Intelligence 22, 5–27 (1998)
2. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2001)
3. Cohen, P.R., Levesque, H.J.: Intention Is Choice with Commitment. Artificial Intelligence 42, 213–261 (1990)
4. de Boer, F., Hindriks, K., van der Hoek, W., Meyer, J.-J.Ch.: A Verification Framework for Agent Programming with Declarative Goals. Journal of Applied Logic 5(2), 277–302 (2007)
5. Emerson, E.A.: Temporal and Modal Mogic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B. North-Holland Publishing Company, Amsterdam (1990)
6. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
7. Herzig, A., Longin, D.: C&l intention revisited. In: Proc. of the 9th Int. Conference Principles of Knowledge Representation and Reasoning (KR 2004), pp. 527–535 (2004)
8. Hindriks, K.V., de Boer, F.S., van der Hoek, W., Meyer, J.-J.C.: Agent Programming with Declarative Goals. In: Castelfranchi, C., Lespérance, Y. (eds.) ATAL 2000. LNCS (LNAI), vol. 1986, pp. 228–243. Springer, Heidelberg (2001)
9. Hindriks, K.V., van Riemsdijk, B.: Using temporal logic to integrate goals and qualitative preferences into agent programming. In: Proceedings of the International Workshop on Declarative Agent Languages and Theories (accepted, 2008)
10. Meyer, J.-J.Ch.: Our quest for the holy grail of agent verification. Automated Reasoning with Analytic Tableaux and Related Methods, 2–9 (2007)
11. Rao, A.S., Georgeff, M.P.: Intentions and Rational Commitment. Technical Report 8, Australian Artificial Intelligence Institute (1993)
12. van der Hoek, W., Wooldridge, M.: Towards a Logic of Rational Agency. Logic Journal of the IGPL 11(2), 133–157 (2003)

Linear Exponentials as Resource Operators: A Decidable First-order Linear Logic with Bounded Exponentials

Norihiro Kamide

Waseda Institute for Advanced Study,
1-6-1 Nishi Waseda, Shinjuku-ku, Tokyo 169-8050, Japan
logician-kamide@aoni.waseda.jp

Abstract. It is known that Girard's linear logics can elegantly represent the concept of "resource consumption". The linear exponential operator ! in linear logics can express a specific infinitely reusable resource (i.e., it is reusable not only for any number, but also many times). It is also known that the propositional intuitionistic linear logic with ! and the first-order intuitionistic linear logic without ! (called here ILL) are undecidable and decidable, respectively. In this paper, a new decidable first-order intuitionistic linear logic, called the resource-indexed linear logic RL[l], is introduced by extending and generalizing ILL. The logic RL[l] has an l-bounded exponential operator $!_l$, and this operator can express a specific finitely usable resource (i.e., it is usable in any positive number less than $l+1$, but only once). The embedding theorem of RL[l] into ILL is proved, and by using this theorem, the cut-elimination and decidability theorems for RL[l] are shown.

1 Introduction

1.1 Motivations and Aims

The notion of "resource", encompassing concepts such as processor time, memory, cost of components and energy requirements, is fundamental to computational systems [13]. In the area of AI, this notion is also very important in handling real scheduling problems to construct complex plans of actions, since many actions consume resources, such as money, gas and raw materials [14] (see Section 12 in [14]). An approach towards a logical theory of resources has been developed by Pym, O'Hearn and Yang [13], using the logic BI of bunched implications, which is a combination of both the multiplicative fragment of intuitionistic linear logic and the intuitionistic logic. The present paper's approach is regarded as an alternative to such an approach.

It is known that Girard's *linear logics* can elegantly represent the concept of "resource consumption" [2]. The linear exponential operator ! in linear logics can express a specific infinitely reusable resource (i.e., it is reusable not only for any number, but also many times). In this respect, however, there is no infinite resource in the real world, and hence the original exponential operator is considered

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 245–257, 2008.

to be inappropriate in order to formalize realistic resource-sensitive reasoning. Formalizing such realistic *finite* resource-aware reasoning needs a modified or restricted version of the original operator.

The central aim of this paper is to express a more appropriate (i.e., finite) and fine-grained form of resource-sensitive reasoning based on a new decidable first-order intuitionistic linear logic, called the *resource-indexed linear logic* RL[l]. The proposed logic RL[l] has an l-bounded exponential operator $!_l$, and this operator can represent a specific finitely usable resource (i.e., it is usable in any positive number less than $l+1$, but only once). An example of such a resource is a computer virus or vaccine program. For example, an expression "$!_l$ virus" in RL[l] means "The virus program can be scattered in a number less than $l+1$, but only once (i.e., it is consumed after use)."

1.2 Linear Exponentials as Resource Operators

A characteristic inference rule for the original exponential operator ! in Girard's classical and intuitionistic linear logics is:

$$\frac{!\alpha, !\alpha, \Gamma \Rightarrow \gamma}{!\alpha, \Gamma \Rightarrow \gamma} \ ,$$

which is regarded as a modal refinement of the contraction rule:

$$\frac{\alpha, \alpha, \Gamma \Rightarrow \gamma}{\alpha, \Gamma \Rightarrow \gamma} \ .$$

This characteristic rule corresponds to the Hilbert-style axiom scheme $!\alpha \rightarrow !\alpha * !\alpha$. The intended meaning of the formula of the form $!\alpha$ is then:

- "The resource α is reusable in any number and many times (i.e., it is reusable as many times as needed),"

The soft exponential operator $!_s$ in Lafont's *soft linear logic* (SLL) [10] is useful for representing specific infinitely usable resources [4]. The operator $!_s$ is characterized by the multiplexing rule:

$$\frac{\overbrace{\alpha, ..., \alpha}^{n}, \Gamma \Rightarrow \gamma}{!_s\alpha, \Gamma \Rightarrow \gamma}$$

where n can be any natural number. This rule corresponds to the Hilbert-style axiom scheme $!_s\alpha \rightarrow \mathbf{1} \wedge \overbrace{\alpha * \alpha * \cdots * \alpha}^{n}$, and hence represents the infinite (onetime) usability of a resource formula α (i.e., it is consumed after use). The intended meaning of the formula of the form $!_s\alpha$ is:

- "The resource α is usable in any number, but only once (i.e., it is not reusable many times, but usable any number)."

The bounded soft exponential operator $!_r$ in *bounded soft linear logic* (BSLL) [5], which is a restricted verson of $!_s$, is characterized by the bounded multiplexing rule:

$$\frac{\overbrace{\alpha, ..., \alpha}^{0 \leq n \leq r}, \Gamma \Rightarrow \gamma}{!_r\alpha, \Gamma \Rightarrow \gamma}$$

where r is a fixed positive integer. The intended meaning of the formula of the form $!_r\alpha$ is:

- "The resource α is usable in any *finite* number less than $r+1$, but only once (i.e., it is consumed after use)".

It is remarked that the idea of "bounded exponentials" is not a new idea. Such an idea was first developed by Girard et al. [3] in order to characterize polynomial-time computation by *bounded linear logic* (BLL). The original bounded exponential $!_b$ proposed by them are different from the bounded soft exponential $!_r$ and the proposed bounded exponential $!_l$ in this paper. The bounded exponential $!_b$ of BLL was based on the notion of *resource polynomials*, and the inference rules for $!_b$ were quite different from those for $!_r$ and $!_l$. The bounded exponential rules for $!_b$ have the same forms as in the original linear logic such as the !-contraction and the !-dereliction, and these rules are modified by putting the resource polynomial bound. We cannot obtain a simple intuitive interpretation for $!_b$ because of the complexity of the setting of the inference rules for $!_b$.

1.3 Proposed Approach

In the present paper, a modified new l-bounded exponential operator $!_l$, which is useful for representing specific finitely usable resources, is introduced. The proposed new logic RL[l] is obtained from the first-order intuitionistic linear logic by replacing ! by $!_l$ and by generalizing initial sequents and inference rules by putting explicit resource indexes.

The operator $!_l$ is characterized by the following inference rules: for a fixed positive integer l,

$$\frac{\overbrace{\alpha * \cdots * \alpha}^{1 \leq n \leq l}, \Gamma \Rightarrow \gamma}{!_l\alpha, \Gamma \Rightarrow \gamma} \qquad \frac{\Gamma \Rightarrow \alpha \quad \Gamma \Rightarrow \alpha * \alpha \quad \cdots \quad \Gamma \Rightarrow \overbrace{\alpha * \cdots * \alpha}^{l}}{\Gamma \Rightarrow\ !_l\alpha}$$

which correspond to the Hilbert-style axiom scheme:

$$!_l\alpha \leftrightarrow \alpha \wedge (\alpha * \alpha) \wedge (\alpha * \alpha * \alpha) \wedge \cdots \wedge (\overbrace{\alpha * \cdots * \alpha}^{l}).$$

Since the intended meaning of the formula of the form $\overbrace{\alpha * \cdots * \alpha}^{i}$ is:

- "The resource α is usable *just* in the number i, but only once", the intended meaning of the formula of the form $!_l\alpha$ is[1]:

[1] We do not determine the rigid expression of the following sentence using $!_l$: "α is usable in any positive number [of times] less than 5." The following modified sentence may be expressed as $(!_4\alpha)^2$: "α is usable in any positive number [of times] less than 5, but only twice".

- "The resource α is usable in any finite positive number less than $l+1$, but only once (i.e., it is consumed after use)."

A technical merit of introducing $!_l$ is that the decidability of RL[l] can be shown.

1.4 Decision Problems in Linear Logics

It is known that the modal propositional intuitionistic linear logic with ! and the non-modal first-order intuitionistic linear logic without ! (called here ILL) are undecidable and decidable, respectively [2,6,7,8,11,12].

A brief history of decision problems in linear logics is explained below. It is known that the first-order classical and intuitionistic logics are undecidable, but some sublogics of these first-order logics by deleting the contraction rule(s) are decidable [1,6,8,7,12]. A decidable sublogic of the first-order classical logic was studied by Ketonen and Weyhrauch [6]. Such a logic, called by them *direct predicate calculus*, has indeed no contraction rule. A similar setting of decidable contraction-free first-order logic was studied by Mey [12]. A more detailed decision procedure for the direct predicate calculus was studied by Bellin and Ketonen from the point of view of proof nets and implementation [1]. Some sublogics of the first-order intuitionistic logics by deleting the contraction rule were studied by Komori [8]. He showed that some first-order intuitionistic substructural logics without the contraction rule, including ILL, are decidable. A role of the contraction rule in decision problems for sublogics of the intuitionistic logic was clarified by Kiriyama and Ono [7]. They showed that ILL is still decidable in the language with function symbols.

In comparison with the non-modal first-order cases without !, "modal" propositional cases with ! has a different situation. Decision problems for propositional linear logics were comprehensively studied by Lincoln et al. [11]. It was shown in [11] that the modal propositional classical and intuitionistic linear logics with ! are undecidable. It was shown by Kopylov [9] that the modal propositional affine linear logic with ! (i.e., the logic has weakening) is decidable. Although the decision problem for the propositional SLL with $!_s$ has not been solved yet, the propositional BSLL with $!_r$ was shown to be decidable [5]. It is unknown whether the first-order version of BSLL is decidable or not.

Some simple questions, which are motivated to extend ILL in order to express fine-grained and expressive resource-aware reasoning more appropriately, are now arisen: "Can a decidable first-order linear logic with restricted exponentials be constructed?" The result of this paper is regarded as an answer of this question: The logic RL[l], which is a natural extension and generalization of ILL by adding a finite l-bounded version $!_l$ of !, is shown to be decidable. The decidability of RL[l] is proved by using an embedding of RL[l] into ILL.

1.5 Summary of This Paper

In Section 2, the resource-indexed linear logic RL[l] and its infinite version RL[ω] are introduced as sequent calculi, and an admissible sequent and some provable sequents are addressed.

In Section 3, firstly, the embedding theorem of RL[ω] into ILL is proved, and then, the cut-elimination and decidability theorems of RL[ω] are shown using the embedding theorem.

In Section 4, some extensions of RL[l], such as an extended RL[l] with the "why not" version of $!_l$ and the original exponential operator !, are presented. The (modified) embedding and cut-elimination theorems can also be shown for these extensions.

In Section 5, this paper is concluded, and some remarks are given.

2 Sequent Calculus

Prior to the precise discussion, the language $\mathcal{L}$ used for RL[l] is introduced as follows: $\mathcal{L}$ consists of individual variables, (n-ary) predicate symbols, $\mathbf{1}$ (multiplicative truth constant), $\bot$ (additive falsity constant), $\rightarrow$ (implication), $\wedge$ (conjunction), $*$ (fusion), $\vee$ (disjunction), $\forall$ (for all), $\exists$ (exists) and $!_l$ (l-bounded exponential operator or l-bounded "of course" operator). Small letters x, y, z, ... are used for individual variables, small letters p, q,... are used for (n-ary) predicate symbols or atomic formulas, Greek small letters $\alpha, \beta, ...$ are used for formulas, and Greek capital letters $\Gamma, \Delta, ...$ are used for finite (possibly empty) multisets of formulas. The symbol ω is used to represent the set of natural numbers. The symbol ω^+ is used to represent the set of positive integers. Let l be a fixed finite positive integer. Then, the symbol ω_l is used to represent the set $\{i \in \omega \mid i < l\}$. Lower-case letters i, j and k are sometimes used to denote any positive integers. An expression α^i for any $i \in \omega^+$ is used to denote $\overbrace{\alpha * \alpha * \cdots * \alpha}^{i}$. In this expression, the superscript $\cdot^i$ of α is called the *resource index* of α. The intended meaning of α^i is thus "The resource α is usable just in the number i, but only once." The symbol $\equiv$ means the equality of sequences (or multisets) of symbols. A *sequent* is an expression of the form $\Gamma \Rightarrow \gamma$ (the succedent of the sequent is non empty). If a sequent S is provable in a sequent calculus L, then such a fact is denoted as $L \vdash S$ or $\vdash S$. The parentheses for $*$ is omitted since $*$ is associative, i.e., $\vdash \alpha * (\beta * \gamma) \Rightarrow (\alpha * \beta) * \gamma$ and $\vdash (\alpha * \beta) * \gamma \Rightarrow \alpha * (\beta * \gamma)$ for any formulas α, β and γ. Since all logics discussed in this paper are formulated as sequent calculi, a sequent calculus will occasionally be identified with the logic determined by it.

A *resource-indexed linear logic* RL[l] with a resource bound l is defined below.

Definition 1 (RL[l]). *An expression α^i with $i \in \omega^+$ is inductively defined by ($\alpha^1 := \alpha$) and ($\alpha^{i+1} := \alpha^i * \alpha$). Let l be a fixed positive integer (called a* resource bound*).*

The initial sequents of RL[l] *are of the form: for any atomic formula p and any $i \in \omega^+$,*

$$p^i \Rightarrow p^i \qquad \Rightarrow \mathbf{1}^i \qquad \bot^i, \Gamma \Rightarrow \gamma.$$

The cut rule of RL[l] *is of the form: for any $i \in \omega^+$,*

$$\frac{\Gamma \Rightarrow \alpha^i \quad \alpha^i, \Delta \Rightarrow \gamma}{\Gamma, \Delta \Rightarrow \gamma}\ (\text{cut}).$$

The logical inference rules of RL[l] *are of the form: for any* $i \in \omega^+$,

$$\frac{\Gamma \Rightarrow \gamma}{\mathbf{1}^i, \Gamma \Rightarrow \gamma}\ (\mathbf{1}\mathrm{we})$$

$$\frac{\Gamma \Rightarrow \alpha^i \quad \beta^i, \Delta \Rightarrow \gamma}{(\alpha{\rightarrow}\beta)^i, \Gamma, \Delta \Rightarrow \gamma}\ (\rightarrow\mathrm{left}) \qquad \frac{\alpha^i, \Gamma \Rightarrow \beta^i}{\Gamma \Rightarrow (\alpha{\rightarrow}\beta)^i}\ (\rightarrow\mathrm{right})$$

$$\frac{\alpha^i, \beta^i, \Gamma \Rightarrow \gamma}{(\alpha * \beta)^i, \Gamma \Rightarrow \gamma}\ (*\mathrm{left}) \qquad \frac{\Gamma \Rightarrow \alpha^i \quad \Delta \Rightarrow \beta^i}{\Gamma, \Delta \Rightarrow (\alpha * \beta)^i}\ (*\mathrm{right})$$

$$\frac{\alpha^i, \Gamma \Rightarrow \gamma}{(\alpha \wedge \beta)^i, \Gamma \Rightarrow \gamma}\ (\wedge\mathrm{left1}) \qquad \frac{\beta^i, \Gamma \Rightarrow \gamma}{(\alpha \wedge \beta)^i, \Gamma \Rightarrow \gamma}\ (\wedge\mathrm{left2})$$

$$\frac{\Gamma \Rightarrow \alpha^i \quad \Gamma \Rightarrow \beta^i}{\Gamma \Rightarrow (\alpha \wedge \beta)^i}\ (\wedge\mathrm{right}) \qquad \frac{\alpha^i, \Gamma \Rightarrow \gamma \quad \beta^i, \Gamma \Rightarrow \gamma}{(\alpha \vee \beta)^i, \Gamma \Rightarrow \gamma}\ (\vee\mathrm{left})$$

$$\frac{\Gamma \Rightarrow \alpha^i}{\Gamma \Rightarrow (\alpha \vee \beta)^i}\ (\vee\mathrm{right1}) \qquad \frac{\Gamma \Rightarrow \beta^i}{\Gamma \Rightarrow (\alpha \vee \beta)^i}\ (\vee\mathrm{right2})$$

$$\frac{\alpha^i[y/x], \Gamma \Rightarrow \gamma}{(\forall x\alpha)^i, \Gamma \Rightarrow \gamma}\ (\forall\mathrm{left}) \qquad \frac{\Gamma \Rightarrow \alpha^i[z/x]}{\Gamma \Rightarrow (\forall x\alpha)^i}\ (\forall\mathrm{right})$$

$$\frac{\alpha^i[z/x], \Gamma \Rightarrow \gamma}{(\exists x\alpha)^i, \Gamma \Rightarrow \gamma}\ (\exists\mathrm{left}) \qquad \frac{\Gamma \Rightarrow \alpha^i[y/x]}{\Gamma \Rightarrow (\exists x\alpha)^i}\ (\exists\mathrm{right})$$

where $\alpha^i[z/x]$ *(*$\alpha^i[y/x]$*) is the formula obtained from* α^i *by replacing all free occurrences of* x *in* α^i *by an arbitrary individual variable* z *not occurring in the lower sequent (an arbitrary individual variable* y*, respectively), but avoiding the clash of variables.*

The l*-bounded exponential inference rules of* RL[l] *are of the form: for any* $i \in \omega^+$,

$$\frac{\alpha^{i+k}, \Gamma \Rightarrow \gamma}{(!_l\alpha)^i, \Gamma \Rightarrow \gamma}\ (!_l\mathrm{left}) \qquad \frac{\{\ \Gamma \Rightarrow \alpha^{i+j}\ \}_{j\in\omega_l}}{\Gamma \Rightarrow (!_l\alpha)^i}\ (!_l\mathrm{right})$$

where $k \in \omega_l$.

Definition 2 (RL[ω]). RL[ω] *is obtained from* RL[l] *by replacing* ($!_l$left) *and* ($!_l$right) *by the unbounded exponential inference rules of the form: for any* $n \in \omega$ *and any* $i \in \omega^+$,

$$\frac{\alpha^{i+n}, \Gamma \Rightarrow \gamma}{(!_\omega\alpha)^i, \Gamma \Rightarrow \gamma}\ (!_\omega\mathrm{left}) \qquad \frac{\{\ \Gamma \Rightarrow \alpha^{i+j}\ \}_{j\in\omega}}{\Gamma \Rightarrow (!_\omega\alpha)^i}\ (!_\omega\mathrm{right}).$$

Definition 3 (ILL). *A sequent calculus* ILL *for Girard's non-modal first-order intuitionistic linear logic is obtained from* RL[l] *by deleting* ($!_l$left), ($!_l$right), *and replacing* i *by* 1. *The modified inference rules for* ILL *by replacing* i *by* 1 *are denoted by labelling "ILL" in superscript, e.g.,* ($\rightarrow\mathrm{left}^{ILL}$). *It is noted that* ILL *is a special case of the* $!_l$*-free fragment of* RL[l].

It is remarked that the exchange rule is omitted in RL[l], since multisets as the antecedents of sequents are adopted. It is also remarked that for any formula α, the sequent of the form $\alpha^i \Rightarrow \alpha^i$ is provable in RL[l]. This can be shown by induction on α. Thus, the sequents of the form $\alpha^i \Rightarrow \alpha^i$ can also be regarded as the initial sequents of RL[l].

Proposition 4. *The* $!_l$*-free fragment* RL *of* RL[l] *is strictly stronger than* ILL.

Proof. For an atomic formula p, the sequent $p^2 \to p^2 \Rightarrow (p \to p)^2$ is not provable in ILL, but it is provable in RL. ∎

It is remarked that RL[l] is just a logic parameterized by a fixed concrete positive integer l. Thus, before the detailed discussion, we have to fix RL[l] as a concrete logic such as RL[5]. Indeed, for example, RL[3] is different from RL[2]: $p \wedge p^2 \Rightarrow\ !_2 p$ is provable in RL[2], but $p \wedge p^2 \Rightarrow\ !_3 p$ is not provable in RL[3]. The unprovability of the sequent is due to the cut-elimination theorem for RL[l] (Theorem 10), which theorem will be proved in a later section.

Proposition 5. *Let* m *and* n *be distinct fixed positive integers. The logics* RL[m] *and* RL[n] *are not theorem-equivalent.*

Proof. By using Theorem 10. ∎

The specific inference rules ($!_l$left) and ($!_l$right) in RL[l] characterize the following Hilbert-style axiom scheme: $!_l\alpha \leftrightarrow (\alpha \wedge \alpha^2 \wedge \cdots \wedge \alpha^l)$, and hence the intended meaning of $!_l\alpha$ is "The resource α is usable in any number less than $l+l$, but only once." The unbounded inference rules ($!_\omega$left) and ($!_\omega$right) in RL[ω] characterize the following Hilbert-style axiom scheme: $!_\omega\alpha \leftrightarrow (\alpha \wedge \alpha^2 \wedge \alpha^3 \wedge \cdots \infty)$, and hence the intended meaning of $!_\omega\alpha$ is "The resource α is usable in any number, but only once." It is noted that the inference rule ($!_\omega$left) is similar to the multiplexing rule of the soft exponential operator $!_s$ in Lafont's SLL. Since the treatment of the infinite premises rule ($!_\omega$right) is somewhat difficult, we do not know whether RL[ω] is decidable or not. Such a problem is remained as an open question. In this paper, we do not discuss more about RL[ω], since the aim of this paper is to obtain a decidable first-order linear logic.

Some admissible rules and provable sequents in RL[l] are presented in Propositions 6 and 7. These propositions imply that every resource index $\cdot^i$ with $i \geq 2$ is ragarded as a modal operator stronger than that of the modal logic K.

Proposition 6. *An expression* Γ^i *means the multiset* $\{\gamma^i \mid \gamma \in \Gamma\}$. *The rules of the form: for any* $i \in \omega^+$ *with* $i \geq 2$,

$$\frac{\Gamma \Rightarrow \gamma}{\Gamma^i \Rightarrow \gamma^i}\ (\cdot^i \text{ regu})$$

are admissible in cut-free RL[l].

Proposition 7. *An expression* $\alpha \Leftrightarrow \beta$ *means the sequents* $\alpha \Rightarrow \beta$ *and* $\beta \Rightarrow \alpha$. *The following sequents are provable in* RL[l]*: for any formulas* α, β *and any* $i \in \omega^+$,

(1) $\mathbf{1}^i \Leftrightarrow \mathbf{1}$,
(2) $\bot^i \Leftrightarrow \bot$,
(3) $(\alpha \circ \beta)^i \Leftrightarrow \alpha^i \circ \beta^i$ *where* $\circ \in \{\rightarrow, \wedge, *, \vee\}$,
(4) $(\sharp x\alpha)^i \Leftrightarrow \sharp x(\alpha^i)$ *where* $\sharp \in \{\forall, \exists\}$,
(5) $(!_l\alpha)^i \Leftrightarrow\ !_l(\alpha^i)$,
(6) $!_l\alpha \Leftrightarrow \alpha \wedge \alpha^2 \wedge \cdots \wedge \alpha^l$.

3 Embedding, Cut-Elimination and Decidability

An expression like $\bigwedge\{\alpha_i \mid i \in \omega_l\}$ where $\{\alpha_i \mid i \in \omega_l\}$ is a multiset means $\alpha_0 \wedge \alpha_1 \wedge \cdots \wedge \alpha_l$. For example, $\bigwedge\{\alpha, \alpha, \beta\}$ means $\alpha \wedge \alpha \wedge \beta$.

Definition 8. *We fix a countable non-empty set Φ of atomic formulas, and define the sets $\Phi_i := \{p_i \mid p \in \Phi\}$ $(2 \leq i \in \omega)$ and $\Phi_1 := \Phi$ of atomic formulas. The language $\mathcal{L}_{\mathrm{RL}[l]}$ of* RL$[l]$ *is defined by using Φ, $\mathbf{1}, \bot, \rightarrow, \wedge, *, \vee, \forall, \exists$ and $!_l$. The language $\mathcal{L}_{\mathrm{ILL}}$ of* ILL *is defined by using $\bigcup_{i\in\omega^+} \Phi_i$, $\mathbf{1}, \bot, \rightarrow, \wedge, *, \vee, \forall$ and $\exists$.*

A mapping f from $\mathcal{L}_{\mathrm{RL}[l]}$ to $\mathcal{L}_{\mathrm{ILL}}$ is defined by: for any $i \in \omega^+$,

1. $f(p^i) := p_i \in \Phi_i$ *for any* $p \in \Phi$ *(especially,* $f(p) := p \in \Phi_1$*),*
2. $f(\mathbf{1}^i) := \mathbf{1}$,
3. $f(\bot^i) := \bot$,
4. $f((\alpha \circ \beta)^i) := f(\alpha^i) \circ f(\beta^i)$ *where* $\circ \in \{\rightarrow, \wedge, *, \vee\}$,
5. $f((\sharp x\alpha)^i) := \sharp x f(\alpha^i)$ *where* $\sharp \in \{\forall, \exists\}$,
6. $f((!_l\alpha)^i) := \bigwedge\{f(\alpha^{i+j}) \mid j \in \omega_l\}$.

An expression $f(\Gamma)$ denotes the result of replacing every occurrence of a formula α in Γ by an occurrence of $f(\alpha)$.

Strictly speaking, the embedding function f is strongly depend on the resource bound l, i.e., f should be denoted as f_l. Indeed, $f_3(!_3p)$ and $f_5(!_5p)$ are different. But, for the sake of brevity, a simple expression f will be used in the following.

Theorem 9 (Embedding). *Let Γ be a multiset of formulas in $\mathcal{L}_{\mathrm{RL}[l]}$, γ be a formula in $\mathcal{L}_{\mathrm{RL}[l]}$, and f be the mapping defined in Definition 8.*

(1) RL$[l] \vdash \Gamma \Rightarrow \gamma$ *iff* ILL $\vdash f(\Gamma) \Rightarrow f(\gamma)$.
(2) RL$[l] - (\mathrm{cut}) \vdash \Gamma \Rightarrow \gamma$ *iff* ILL $- (\mathrm{cut}^{ILL}) \vdash f(\Gamma) \Rightarrow f(\gamma)$.

Proof. Since the case (2) can be obtained as the subproof of the case (1), we show only (1).

$(\Longrightarrow)$: By induction on a proof P of $\Gamma \Rightarrow \gamma$ in RL$[l]$. We distinguish the cases according to the last inference of P, and show some cases.

Case $(p^i \Rightarrow p^i)$: The last inference of P is of the form: $p^i \Rightarrow p^i$. In this case, we obtain $f(p^i) \Rightarrow f(p^i)$, i.e., $p_i \Rightarrow p_i$ $(p_i \in \Phi_i)$. This is an initial sequent of ILL.

Case ($\rightarrow$left): The last inference of P is of the form:

$$\frac{\Gamma \Rightarrow \alpha^i \quad \beta^i, \Delta \Rightarrow \gamma}{(\alpha{\rightarrow}\beta)^i, \Gamma, \Delta \Rightarrow \gamma}\ (\rightarrow\text{left}).$$

By the hypothesis of induction, we have ILL $\vdash f(\Gamma) \Rightarrow f(\alpha^i)$ and ILL $\vdash f(\beta^i)$, $f(\Delta) \Rightarrow f(\gamma)$. Then, we obtain

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ f(\Gamma) \Rightarrow f(\alpha^i) & f(\beta^i), f(\Delta) \Rightarrow f(\gamma)\end{array}}{f(\alpha^i) \rightarrow f(\beta^i), f(\Gamma), f(\Delta) \Rightarrow f(\gamma)}\ (\rightarrow\text{left}^{ILL})$$

where $f(\alpha^i) \rightarrow f(\beta^i) = f((\alpha \rightarrow \beta)^i)$.

Case ($!_l$left): The last inference of P is of the form: for any $k \in \omega_l$,

$$\dfrac{\alpha^{i+k}, \Gamma \Rightarrow \gamma}{(!_l\alpha)^i, \Gamma \Rightarrow \gamma}\ (!_l\text{left}).$$

By the hypothesis of induction, we have ILL $\vdash f(\alpha^{i+k}), f(\Gamma) \Rightarrow f(\gamma)$, and hence obtain:

$$\dfrac{\begin{array}{c}\vdots \\ f(\alpha^{i+k}), f(\Gamma) \Rightarrow f(\gamma) \\ \vdots\ (\wedge\text{left}^{ILL}) \end{array}}{\bigwedge\{f(\alpha^{i+j}) \mid j \in \omega_l\}, f(\Gamma) \Rightarrow f(\gamma)}$$

where $\bigwedge\{f(\alpha^{i+j}) \mid j \in \omega_l\} = f((!_l\alpha)^i)$, and $f(\alpha^{i+k})$ is in the multiset $\{f(\alpha^{i+j}) \mid j \in \omega_l\}$.

Case ($!_l$right): The last inference of P is of the form:

$$\dfrac{\{\ \Gamma \Rightarrow \alpha^{i+j}\ \}_{j \in \omega_l}}{\Gamma \Rightarrow (!_l\alpha)^i}\ (!_l\text{right}).$$

By the hypothesis of induction, we have ILL $\vdash f(\Gamma) \Rightarrow f(\alpha^{i+j})$ for all $j \in \omega_l$. Let Φ be the multiset $\{f(\alpha^{i+j}) \mid j \in \omega_l\}$. We obtain

$$\dfrac{\begin{array}{c}\vdots \\ \{\ f(\Gamma) \Rightarrow f(\alpha^{i+j})\ \}_{f(\alpha^{i+j}) \in \Phi} \\ \vdots\ (\wedge\text{right}^{ILL}) \end{array}}{f(\Gamma) \Rightarrow \bigwedge \Phi}$$

where $\bigwedge \Phi = f((!_l\alpha)^i)$.

($\Longleftarrow$) : By induction on a proof Q of $f(\Gamma) \Rightarrow f(\gamma)$ in ILL. We distinguish the cases according to the last inference of Q, and show only the following case.

Case ($\wedge\text{right}^{ILL}$): The last inference of Q is of the form:

$$\dfrac{f(\Gamma) \Rightarrow f(\alpha^i) \quad f(\Gamma) \Rightarrow f(\beta^i)}{f(\Gamma) \Rightarrow f((\alpha \wedge \beta)^i)}\ (\wedge\text{right}^{ILL})$$

where $f((\alpha \wedge \beta)^i) = f(\alpha^i) \wedge f(\beta^i)$. By the hypothesis of induction, we have RL$[l]$ $\vdash \Gamma \Rightarrow \alpha^i$ and RL$[l] \vdash \Gamma \Rightarrow \beta^i$. Then, we obtain

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ \Gamma \Rightarrow \alpha^i & \Gamma \Rightarrow \beta^i\end{array}}{\Gamma \Rightarrow (\alpha \wedge \beta)^i}\ (\wedge\text{right}).$$

∎

Using this theorem, we can prove the following.

Theorem 10 (Cut-elimination). *The rule* (cut) *is admissible in cut-free* RL[l].

Proof. Suppose RL[l] $\vdash \Gamma \Rightarrow \gamma$. Then, we have ILL $\vdash f(\Gamma) \Rightarrow f(\gamma)$ by Theorem 9 (1), and hence ILL $-$ (cut^{ILL}) $\vdash f(\Gamma) \Rightarrow f(\gamma)$ by the well-known cut-elimination theorem for ILL. By Theorem 9 (2), we obtain RL[l] $-$ (cut) $\vdash \Gamma \Rightarrow \gamma$. ∎

In this paper, the cut-elimination theorem for RL[l] is proved by the way via the embedding theorem. The directed syntactical cut-elimination proof of RL[l] may be obtained using the standard way of Gentzen.

Theorem 11 (Decidability). RL[l] *is decidable.*

Proof. By Theorem 9, the provability of RL[l] can be transformed into that of ILL. The translation from the formulas of RL[l] to the corresponding formulas of ILL can finitely be performed. Since ILL is decidable, RL[l] is also decidable. ∎

4 Extensions

The logic R[l] can be extended with the addition of various constructors, and the embedding and cut-elimination results can also be shown for such extended logics in a similar way. For the decidability issue, although extensions with the original exponential operator ! are undecidable, a number of !-less extensions are shown to be decidable.

4.1 Adding Function Symbols and Constants

Although in the previous sections, the first-order language does not include function and individual constant symbols, an extended language with these symbols can naturally be adapted for the results of the embedding, cut-elimination and decidability. The decidability of ILL with such an extended language was investigated in [7].

An extended language with the multiplicative falsity constant $\mathbf{0}$ and the additive truth constant $\top$ can also be adapted for the results of the embedding, cut-elimination and decidability. In the case with $\mathbf{0}$, the sequent notion in the underlying sequent calculus must be changed to allowing an empty multiset in the succedent, and the following initial sequents and inference rules must be added: for any $i \in \omega^+$,

$$\mathbf{0}^i \Rightarrow \qquad \frac{\Gamma \Rightarrow}{\Gamma \Rightarrow \mathbf{0}^i}\ .$$

4.2 Adding Bounded "Why Not" Operator

It is remarked that the "why not" version $?_l$ of $!_l$ (i.e., $?_l$ is the dual of $!_l$) can be introduced. The inference rules for $?_l$ are of the form: for any $i \in \omega^+$ and any $k \in \omega_l$,

$$\frac{\{\ \alpha^{i+j}, \Gamma \Rightarrow \gamma\ \}_{j \in \omega_l}}{(?_l\alpha)^i, \Gamma \Rightarrow \gamma} \qquad \frac{\Gamma \Rightarrow \alpha^{i+k}}{\Gamma \Rightarrow (?_l\alpha)^i}$$

which correspond to the Hilbert-style axiom schemes: $?_l\alpha \leftrightarrow (\alpha \vee \alpha^2 \vee \cdots \vee \alpha^l)$. The embedding, cut-elimination and decidability theorems for the underlying extended system can be shown with some obvious modifications.

4.3 Adding Original Exponentials

It is remarked that the framework of RL[l] can be combined with the original exponential operator !. For this case, the following inference rules are introduced: for any $i, j \in \omega^+$,

$$\frac{\alpha^i, \Gamma \Rightarrow \gamma}{(!\alpha)^i, \Gamma \Rightarrow \gamma} \quad \frac{(!\Gamma)^i \Rightarrow \alpha^j}{(!\Gamma)^i \Rightarrow (!\alpha)^j} \quad \frac{(!\alpha)^i, (!\alpha)^i, \Gamma \Rightarrow \gamma}{(!\alpha)^i, \Gamma \Rightarrow \gamma} \quad \frac{\Gamma \Rightarrow \gamma}{(!\alpha)^i, \Gamma \Rightarrow \gamma}$$

where $(!\Gamma)^i$ means the multiset $\{(!\gamma)^i \mid \gamma \in \Gamma\}$. The results of the embedding (into ILL with !), cut-elimination and undecidability can be obtained for the extended logic.

4.4 Adding Strong Negation

An extended language with the strong negation connective $\sim$ can be considered. The extended ILL with $\sim$, called here WILL, was introduced and studied by Wansing [15]. By using a similar way, the logic WILL can be generalized and extended to the indexed version RWL[l]. For example, the following inference rules for $\sim$ are adopted for RWL[l]: for any $i \in \omega^+$,

$$\frac{(\sim\alpha)^i, \Gamma \Rightarrow \gamma \quad (\sim\beta)^i, \Gamma \Rightarrow \gamma}{(\sim(\alpha \wedge \beta))^i, \Gamma \Rightarrow \gamma} \quad \frac{(\sim\alpha)^i, \Gamma \Rightarrow \gamma}{(\sim(\alpha \wedge \beta))^i, \Gamma \Rightarrow \gamma} \quad \frac{(\sim\beta)^i, \Gamma \Rightarrow \gamma}{(\sim(\alpha \wedge \beta))^i, \Gamma \Rightarrow \gamma}$$

$$\frac{\alpha^i, \Gamma \Rightarrow \gamma}{(\sim\sim\alpha)^i, \Gamma \Rightarrow \gamma} \quad \frac{\Gamma \Rightarrow \alpha^i}{\Gamma \Rightarrow (\sim\sim\alpha)^i} \quad \frac{\{\ (\sim\alpha)^{i+j}, \Gamma \Rightarrow \gamma\ \}_{j \in \omega_l}}{(\sim!_l\alpha)^i, \Gamma \Rightarrow \gamma} \quad \frac{\Gamma \Rightarrow (\sim\alpha)^{i+k}}{\Gamma \Rightarrow (\sim!_l\alpha)^i}\ .$$

The embedding (into WILL), cut-elimination and decidability theorems can be shown for RWL[l].

4.5 Adding All Constructors

The extended logic with function symbols, $\mathbf{0}$, $\top$, $?_l$, ! and $\sim$ can be constructed naturally, and the corresponding embedding and cut-elimination theorems can be shown for that logic.

5 Concluding Remarks

In this paper, a new first-order logic, resource-indexed linear logic RL[l], was introduced based on the l-bounded exponential operator $!_l$, and the embedding, cut-elimination and decidability theorems were shown for RL[l].

In the theoretical point of view, the decidability of RL[l] is a novel feature, since the standard ILL with ! is known as undecidable. In the practical point of

view, the operator $!_l$ in RL[l] is useful for representing "finite" and "fine-grained" resource-sensitive reasoning appropriately.

It is remarked that the proposed operators $!_l$ and $!_\omega$ can be used to express some specific programs, such as computer virus and vaccine programs. Such a program like virus is, roughly speaking, executable or usable simultaneously in any number, but only once (i.e., not reusable many times, or only one executable). Indeed, any existing old virus and vaccine programs are regarded as unavailable many times. These programs and software may be appropriately expressed using $!_l$ and $!_\omega$. The original exponential ! is interpreted as the "unbounded replication operator" for programs or processes in concurrency theory. In contrast, the restricted exponentials $!_l$ and $!_\omega$ can be interpreted as the "bounded (onetime) replication operators".

We hope that the new exponentials are applicable in a wide range of real situations concerning resource-aware reasoning. We now show such an example below. We consider that a real vending machine can deal with less than 51 cans of juice. Such a situation is briefly expressed as (1): $\vdash coin \Rightarrow juice$, (2): $\vdash coin^n \Rightarrow !_{50} juice$ $(1 \leq n < 51)$ and (3): $\vdash coin^{51} \ * \ !_\omega coin \Rightarrow break$, where (1) means "if we put just one coin in a vending machine, then we can get exactly one cans of juice from the machine", (2) means "we can get 50 cans of juice" and (3) means "if we put the coins more than 50 simultaneously, then the vending machine breaks".

Finally in this paper, it is remarked that RL[ω] is analogous to the well-known linear-time temporal logic LTL. The index $\cdot^i$ and the operator $!_\omega$ in RL[ω] respectively resemble to the next-time operator X and the globally operator G in LTL. Indeed, the formula Gα in LTL can informally be interpreted as

$$\mathrm{G}\alpha \leftrightarrow \alpha \wedge \mathrm{X}\alpha \wedge \mathrm{X}^2\alpha \wedge \mathrm{X}^3\alpha \wedge \cdots \infty$$

where X^i means $\overbrace{\mathrm{X}\mathrm{X}\cdots\mathrm{X}}^{i}$. This interpretation just corresponds to $!_\omega\alpha \leftrightarrow (\alpha \wedge \alpha^2 \wedge \alpha^3 \wedge \cdots \infty)$ in RL[ω]. Although the propositional LTL and the first-order LTL are known as decidable and undecidable, respectively, the decision problems for the propositional RL[ω] and the first-order RL[ω] have not been solved yet.

Acknowledgments. This research was supported by the Alexander von Humboldt Foundation. This work was partially supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B) 20700015, 2008.

References

1. Bellin, G., Ketonen, J.: A decision procedure revisited: Notes on direct logic, linear logic and its implementation. Theoretical Computer Science 95, 115–142 (1992)
2. Girard, J.-Y.: Linear logic. Theoretical Computer Science 50, 1–102 (1987)
3. Girard, J.-Y., Scedrov, A., Scott, P.J.: Bounded linear logic: A modular approach to polynomial-time computability. Theoretical Computer Science 97(1), 1–66 (1992)

4. Kamide, N.: Combining soft linear logic and spatio-temporal operators. Journal of Logic and Computation 14(5), 625–650 (2004)
5. Kamide, N.: Towards a theory of resource: an approach based on soft exponentials. Journal of Applied Non-Classical Logics 17(1), 63–89 (2007)
6. Ketonen, J., Weyhrauch, R.: A decidable fragment of predicate calculus. Theoretical Computer Science 32, 297–307 (1984)
7. Kiriyama, E., Ono, H.: The contraction rule and decision problems for logics without structural rules. Studia Logica 50, 299–319 (1991)
8. Komori, Y.: Predicate logics without the structure rules. Studia Logica 45, 393–404 (1986)
9. Kopylov, A.P.: Decidability of linear affine logic. In: Kozen, D. (ed.) 10th Annual IEEE Symposium on Logic in Computer Science, pp. 496–504 (1995)
10. Lafont, Y.: Soft linear logic and polynomial time. Theoretical Computer Science 318(1-2), 163–180 (2004)
11. Lincoln, P., Mitchell, J., Scedrov, A., Shankar, N.: Decision problems for propositional linear logic. Annals of Pure and Applied Logic 56, 239–311 (1992)
12. Mey, D.: A predicate calculus with control of derivations. In: Börger, E., Kleine Büning, H., Richter, M.M. (eds.) CSL 1989. LNCS, vol. 440, pp. 254–266. Springer, Heidelberg (1990)
13. Pym, D.J., O'Hearn, P.W., Yang, H.: Possible worlds and resources: The semantics of BI. Theoretical Computer Science 315(1), 257–305 (2004)
14. Russell, S., Norvig, P.: Artificial intelligence: A modern approach, 2nd edn. Pearson Education, Inc., London (2003)
15. Wansing, H.: The logic of information structures. In: Wansing, H. (ed.) The Logic of Information Structures. LNCS, vol. 681, pp. 1–163. Springer, Heidelberg (1993)

Fibrational Semantics for Many-Valued Logic Programs: Grounds for Non-Groundness

Ekaterina Komendantskaya[1] and John Power[2,⋆]

[1] INRIA Sophia Antipolis, France
ekaterina.komendantskaya@inria.fr
[2] University of Bath, UK
A.J.Power@bath.ac.uk

Abstract. We introduce a fibrational semantics for many-valued logic programming, use it to define an SLD-resolution for annotation-free many valued logic programs as defined by Fitting, and prove a soundness and completeness result relating the two. We show that fibrational semantics corresponds with the traditional declarative (ground) semantics and deduce a soundness and completeness result for our SLD-resolution algorithm with respect to the ground semantics.

Keywords: Many-valued logic programs, categorical logic, fibrational semantics, ground semantics, SLD-resolution.

1 Introduction

Declarative semantics for logic programming characterises logic programs from the model-theoretic point of view, in particular, it shows a procedure for computing (Herbrand) models of logic programs. Commonly, it is given by defining an appropriate semantic operator that works recursively over the *Herbrand base* and the ground instances of clauses and finally settles on the least Herbrand model of a program, [12]. An assortment of many-valued logic programs has received appropriate declarative semantics: *annotation-free logic programs* [6,7,3,16], *implication-based logic programs* [17], *annotated logic programs* [1,4,8,13]. The declarative semantics received algebraic [4] and categorical [5] account.

Another type of semantics for logic programming is called *operational*. Operational semantics gives a proof-theoretic view on logic programming. Often, it is given by the SLD-resolution, [12]. As for many-valued generalisations of logic programming, the (SLD) resolution procedures were suggested for a number of different many-valued logic programs, [1,8,13,17,3,16].

A third type of semantics, a *fibrational semantics* for logic programming was suggested; [9]. It gave structural (categorical) characterisation of the syntax of logic programs. Unlike declarative semantics, fibrational semantics does not use Herbrand models. As a consequence, this kind of semantics does not depend

⋆ The authors thank the grant "Categorical Semantics for Natural Models of Computation" by the Royal Society/Royal Irish Academy.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 258–271, 2008.

on ground instances of terms, atoms and clauses. Instead, fibrational semantics shows that the syntax of a logic program - sorts of variables, arities of terms, arities of conjunctions in the clause bodies and "←" , - induces a particular structure that characterises the logic program uniquely up to the variable renaming. We will explain this in Section 3. Due to its non-groundness, fibrational semantics can be easily and naturally related to operational semantics and SLD-resolution: neither fibrational, nor operational semantics depend on ground instances of atoms. This is why, the fibrational semantics was used to give a category-theoretic account of SLD-resolution [9,15].

Despite of its elegance, the fibrational semantics has never been extended to any kind of non-classical logic programming. And there was a serious obstacle for such extensions: namely, the fibrational semantics of [9] gave no answer to the question of what role a truth value assignment plays in the new semantics. In fact, this question had no particular importance in case of classical, two valued, logic programs that were analysed in [9], because the evaluation *true* could be automatically assumed for all the clauses constituting a program. And thus, without explicitly mentioning, the fibrational semantics [9] structurally interpreted true unit clauses, and true logical implications between clause bodies and clause heads.

However, in case of many-valued extensions, one cannot simply assume that all the unit clauses are true. Moreover, in case if truth values are not allowed as annotations [6,7], one cannot deduce the truth value of a formula looking simply at the structure of a logic program. Furthermore, it is impossible to assign a truth value to a non-ground formula. In this paper, we analyse this situation categorically. In Section 4 we give a *ground semantics* to many-valued logic programs, respecting the tradition [6,7,4,1] to assign truth values only to ground formulae. In Section 5 we give a fibrational semantics to annotation-free logic programs, and prove that it is equivalent to the ground semantics.

We believe that Proposition 1 and Theorem 1 establishing precise relations between ground and fibrational semantics give theoretical justification for fibrational semantics and break new grounds for future development of the fibrational approach to non-classical logic programming. As an evidence that fibrational semantics can lead to useful applications, we show, in Section 6, that the fibrational semantics for many-valued logic programs gives rise to a novel algorithm of SLD-resolution for annotation-free logic programs. We prove its soundness and completeness relative to the ground and fibrational semantics of Sections 4, 5. In comparison to alternative approaches to many-valued resolution algorithms in [3,16], this novel algorithm provides the ideal compromise between expressiveness and efficiency, as we briefly explain in Section 7.

2 Many-Valued Logic Programs

A conventional (two-valued) logic program [12] consists of a finite set of clauses, some of which form its core, and the rest of which form a database.

Example 1. Let GC (for graph connectivity) denote the logic program with core $(\text{connected}(x,x) \leftarrow)$, $(\text{connected}(x,y) \leftarrow \text{edge}(x,z), \text{connected}(z,y))$.

A database for GC lists the edges of a particular graph: $\text{edge}(a,b) \leftarrow$, $\text{edge}(b,c) \leftarrow$, ….

For the formal analysis of this paper, we need more precision, as follows.

Definition 1. *Given a set $\mathcal{T}$, the set Sort($\mathcal{T}$) of* sorts *generated by $\mathcal{T}$ is the set of all finite, possibly empty, sequences of elements of $\mathcal{T}$.*

We use T_1, T_2 etc., to refer to elements of $\mathcal{T}$; and $\overline{T} = T_1, \ldots, T_n$ to refer to sequences of elements of Sort($\mathcal{T}$). Using categorical notation, we will use the symbol 1 to denote the terminal object (given the empty sequence) in a Cartesian category Sort($\mathcal{T}$), where sequences $T_1, \ldots, T_n$ are seen as finite products. More generally, we will use symbol 1 throughout the paper whenever we talk about an empty product in a given Cartesian category.

Definition 2. *A* sorted language *is a triple $\mathcal{L} = (\mathcal{T}, \mathcal{F}, \mathcal{P})$ consisting of*

- *a set $\mathcal{T}$ of primitive sorts;*
- *for each $\overline{T} \in Sort(\mathcal{T})$ and a primitive sort $T \in \mathcal{T}$, a set $\mathcal{F}(\overline{T}, T)$ of* function symbols *of sort $(\overline{T}, T)$, and*
- *for each $\overline{T} \in Sort(\mathcal{T})$, a set $\mathcal{P}(\overline{T})$ of* predicate symbols *of sort $\overline{T}$.*

Given a sorted language $\mathcal{L} = (\mathcal{T}, \mathcal{F}, \mathcal{P})$ and a set V of variables, we can define terms and atomic formulae as usual, all of these with sorts.

Example 2. The language underlying the logic program from Example 1 is a triple $(\mathcal{T}, \mathcal{F}, \mathcal{P})$ as follows: $\mathcal{T} = \{D\}$; $\mathcal{F}(1, D) = \{a, b, c, \ldots\}$, otherwise $\mathcal{F}(\overline{T}, T)$ is empty; and $\mathcal{P}(DD) = \{\text{connected}, \text{edge}\}$, otherwise $\mathcal{P}(\overline{T})$ is empty.

So, there is one sort D. And there are several nullary function symbols, i.e, constants $a, b, c \ldots$. And there are two binary predicates "connected" and "edge". The sortedness of the predicate amounts simply to their being binary, as the language is single sorted.

Example 3. Suppose we wish to enumerate edges of a given graph using the set of natural numbers. This would require the use of the second sort N. We use predicate "rank" for this purpose. E.g, the clause $(\text{rank}(0, a, b) \leftarrow \text{edge}(a, b))$ describes the basic step of enumeration. Then, we redefine $\mathcal{T}$ of Example 2: $\mathcal{T} = \{D, N\}$; and add $\mathcal{F}(1, N) = \{0, 1, 2, 3, \ldots\}$; and $\mathcal{P}(NDD) = \{\text{rank}\}$. One can use standard predicates "odd" and "even" over natural numbers. Then we would additionally have $\mathcal{P}(N) = \{\text{odd}, \text{even}\}$.

Definition 3. *A* sorted logic program *Γ over the language $\mathcal{L}$ consists of a finite set of clauses $(\overline{T}, \overline{\varphi}, \varphi)$, where $\overline{T}$ is a* sort *of a clause, $\overline{\varphi}$ is a formula of the form $P_1(\overline{t_1}) \wedge \ldots \wedge P_n(\overline{t_n})$ and φ is an atomic formula of the form $P(\overline{t})$; both $\overline{\varphi}$ and φ are of sort $\overline{T}$.*

Example 4. Example 1 is an example of a logic program with one sort. In Example 2, we expressed the language formally, with the sort denoted by D. The logic program has two clauses $(\text{connected}(x,x) \leftarrow)$ and $(\text{connected}(x,y) \leftarrow \text{edge}(x,z), \text{connected}(z,y))$ in its core. They are of sorts D and DDD respectively. Additional clauses $(\text{edge}(a,b) \leftarrow)$, $(\text{edge}(b,c) \leftarrow)$ are of sort 1.

Thus, the sort of a clause depends on the number and sorts of free variables. That is, although the predicate "connected" is binary, the clause $(\text{connected}(a,x) \leftarrow)$ would be of sort D. The clause $(\text{rank}(0,a,b) \leftarrow \text{edge}(a,b))$ from Example 3 would be of sort 1. The clause $(\text{rank}(n+1,x,y) \leftarrow \text{edge}(x,y), \text{rank}(n,z,x))$ would be of sort $NDDD$.

Many-valued annotation-free logic programs [6,7], are formally the same as two-valued logic programs, see Definition 3. But while each atomic ground formula of a two-valued logic program is given an interpretation in $\{0,1\}$, an atomic formula of a many-valued logic program receives an interpretation in an arbitrary specified preorder Ω with finite meets.

Example 5. Our leading example of an annotation-free logic program is as follows. Let Ω be the unit interval $[0,1]$. The logic program of Example 1 is, by definition, also an annotation-free ($[0,1]$-based) logic program. But each ground atom, e.g., $(\text{edge}(a,b))$ or $(\text{connected}(a,b))$, is assigned a truth value from $[0,1]$, (cf. the notion of probabilistic graph, where edges and connections in a graph exist with some probability).

If we have a ground clause $(\text{connected}(a,b) \leftarrow \text{edge}(a,c), \text{connected}(c,b))$, we say that the clause is *true* relative to an interpretation if $|\text{edge}(a,c)| \wedge |\text{connected}(c,b)| \leq |\text{connected}(a,b)|$ in $[0,1]$.

3 The Syntax Viewed through Fibers

In this section we give a fibrational, or equivalently, indexed category based semantics to logic programs. In this section, we consider only the syntax of logic programs, prior to assigning any truth values, and so we essentially rephrase the fibrational semantics outlined in [9]. The reader can find missing definitions and explanations in [14,2,10].

We start by giving a structural interpretation to terms.

Definition 4. *Given $(\mathcal{T},\mathcal{F})$ (before one adds the set $\mathcal{P}$ of predicate symbols) and given a category C with strictly associative finite products, a* pre-interpretation *of $(\mathcal{T},\mathcal{F})$ in C is a function $\gamma : \mathcal{T} \to \mathtt{ob}(C)$ together with, for each function symbol f of sort $(\overline{T},T)$ a map in C from $\gamma(T_1) \times \ldots \times \gamma(T_n)$ to $\gamma(T)$.*

One needs to show that such pre-interpretation exists and that it is unique. (Uniqueness of (pre)-interpretation is synonymous to its minimality in conventional terminology.) This was proved in [9] by constructing the category $C_{\mathcal{T},\mathcal{F}}$ with strictly associative finite products and the unique pre-interpretation $\| \|_{\mathcal{T},\mathcal{F}}$ of $(\mathcal{T},\mathcal{F})$ in $C_{\mathcal{T},\mathcal{F}}$, as follows. The objects of $C_{\mathcal{T},\mathcal{F}}$ are finite sequences of elements of $\mathcal{T}$. An arrow from $\overline{T}$ to T is an equivalence class of terms of arity $T_1 \times \ldots \times T_n$ and type T, i.e, terms are factored out by renaming of variables.

Having interpreted terms, we continue with interpretation for formulae. For this, we need the notion of an indexed category with finite products.

Definition 5. *An* indexed category *over a small category C is a functor $p : C^{op} \to Cat$. An* indexed functor *from p to q is a natural transformation $\tau : p \Rightarrow q : C^{op} \to Cat$.*

Let FP_s be the category of small categories with finite products and functors that strictly preserve finite products.

Definition 6. *If a small category C has finite products, an indexed category $p : C^{op} \to Cat$ has* finite products *if $p : C^{op} \to Cat$ factors through FP_s, i.e, there is a functor $f : C^{op} \to FP_s$ such that $p = U \circ f$, where $U : FP_s \to Cat$ is inclusion.*

An indexed functor $h : p \Rightarrow q$ between indexed categories with finite products respects finite products *if each component does so.*

We say that p has *strictly associative* finite products if C and each $p(X)$ have strictly associative finite products.

We extend the definition of an interpretation of $(\mathcal{T}, \mathcal{F})$ in $C_{\mathcal{T},\mathcal{F}}$ to an interpretation of a language $\mathcal{L} = (\mathcal{T}, \mathcal{F}, \mathcal{P})$ in an indexed category with finite products over $C_{\mathcal{T},\mathcal{F}}$ as follows.

Definition 7. *An* interpretation of a sorted language $\mathcal{L} = (\mathcal{T}, \mathcal{F}, \mathcal{P})$ *in an indexed category $p : C_{\mathcal{T},\mathcal{F}} \to Cat$ with finite products is given by the pre-interpretation $\| \ \|_{\mathcal{T},\mathcal{F}}$: of $(\mathcal{T}, \mathcal{F})$ in $C_{\mathcal{T},\mathcal{F}}$, together with, for each sort $\overline{T} = T_1, \ldots T_n$, a function $\| \ \|_{\mathcal{P}(\overline{T})} : \mathcal{P}(\overline{T}) \to \mathtt{ob}(p(\|T_1\|_{\mathcal{T},\mathcal{F}} \times \ldots \times \|T_n\|_{\mathcal{T},\mathcal{F}}))$.*

Existence and uniqueness of such interpretation was proved in [9]. The free indexed category $p_{\mathcal{L}}$ with strictly associative finite products over $C_{\mathcal{T},\mathcal{F}}$, with an interpretation $\| \ \|_{\mathcal{L}}$ of $\mathcal{L}$ in $p_{\mathcal{L}}$ for a sorted language $\mathcal{L} = (\mathcal{T}, \mathcal{F}, \mathcal{P})$, is given as follows.

* For each $\overline{T} \in \mathtt{ob}(C_{\mathcal{T},\mathcal{F}})$, $p_{\mathcal{L}}(\overline{T})$ is the category with strictly associative finite products freely generated by $(\Phi_{\overline{T}}, \emptyset)$, where $\Phi_{\overline{T}}$ is the set of all triples $(\overline{U}, P, v)$ with $\overline{U} \in \mathrm{Sort}(\mathcal{T})$, a predicate symbol $P \in \mathcal{P}(\overline{U})$ and an arrow $v \in C_{\mathcal{T},\mathcal{F}}(\overline{T}, \overline{U})$. (The symbol $\emptyset$ in $(\Phi_{\overline{T}}, \emptyset)$ indicates that the logic programming arrows "$\leftarrow$" are not interpreted yet. Finite products of triples $(\overline{U}, P, v)$ give account to finite conjunctions.)

** For each $v \in C_{\mathcal{T},\mathcal{F}}(\overline{T}, \overline{U})$, we define the functor $p_{\mathcal{L}}(v) : p_{\mathcal{L}}(\overline{U}) \to p_{\mathcal{L}}(\overline{T})$ by specifying the value of $p_{\mathcal{L}}(v)(\overline{V}, P, s)$, with $s \in C_{\mathcal{T},\mathcal{F}}(\overline{U}, \overline{V})$, to be $(\overline{V}, P, s \circ v)$.

We can identify an object of $p_{\mathcal{L}}(\overline{T})$ with an equivalence class of finite sequences of atomic formulae with free variables of sort $\overline{T}$. We treat the finite sequence as a conjunction.

Definition 8. *Given a logic program Γ over the language $\mathcal{L}$, an* interpretation *of Γ in an indexed category p with strictly associative finite products is given by the following data:*

- *an interpretation* $\| \ \|$ *of* $\mathcal{L}$ *in* p *and*
- *for each sort* $\mathcal{T}$*, formula* $\overline{\varphi}$ *and atomic formula* φ *in* $p(\overline{T})$*, a function* $\| \ \| : \Gamma_{\overline{T}}(\overline{\varphi}, \varphi) \to p(\overline{T})(\|\varphi_1\| \times \ldots \times \|\varphi_n\|, \|\varphi\|)$*, where* $\Gamma_{\overline{T}}(\overline{\varphi}, \varphi)$ *is the family of clauses in* Γ *of the form* $(\overline{T}, \overline{\varphi}, \varphi)$*.*

The existence and uniqueness of such interpretation was proved in [9]. The unique interpretation was called p_Γ, and was essentially $p_\mathcal{L}$, but with added arrows that model the implication arrows "$\leftarrow$".

Example 6. In Example 4, categories $p_\Gamma(1)$, $p_\Gamma(D)$, $p_\Gamma(DD)$, $p_\Gamma(DDD)$, $p_\Gamma(NDDD)$ would be "fibers" generated by clauses of corresponding types.

In the many-valued setting that we will develop in the following sections, our attention will be on indexed category p for which each $p(\overline{T})$ is a preorder Ω with finite meets. In this case, the new "condition" in p_Γ amounts to the assertion that each clause $\varphi \leftarrow \overline{\varphi}$ is sent to an inequality $\|\varphi_1\| \wedge \ldots \wedge \|\varphi_n\| \leq \|\varphi\|$.

4 Ground Semantics, Fibrationally

We first show how the fibrational semantics fits into the framework of traditional declarative (ground) semantics.

We first choose a preorder Ω with finite meets in which to take values. By ground semantics for the underlying language $\mathcal{L}$ we mean the assignment, to each ground formula, of an element of Ω, respecting the structure of $\mathcal{L}$. This amounts to a finite product preserving functor from $p_\mathcal{L}(1)$ to Ω, where the latter is seen as a category with finite products. We extend it to the logic program Γ.

By previous discussions, $C_{\mathcal{T},\mathcal{F}}$ is the category with strictly associative finite products freely generated by $(\mathcal{T}, \mathcal{F})$. Let 1 be the terminal object of $C_{\mathcal{T},\mathcal{F}}$. So, for each $\overline{T} \in \mathtt{ob}(C_{\mathcal{T},\mathcal{F}})$, the homset $C_{\mathcal{T},\mathcal{F}}(1, \overline{T})$ is the set of ground terms of type $\overline{T}$. Moreover, $p_\mathcal{L}(1)$ is the category with strictly associative finite products freely generated by $(\Phi_I, \emptyset)$, with Φ_I being the set of all triples $(\overline{U}, P, v)$, where $v \in C_{\mathcal{T},\mathcal{F}}(1, \overline{U})$. Thus $p_\mathcal{L}(1)$ is the set of all ground formulae of the language $\mathcal{L}$ with finite meets, and it corresponds to the *Herbrand base*. An *interpretation* $| \ |$ of $\mathcal{L}$ in Ω is defined to be a finite meet preserving function from $p_\mathcal{L}(1)$ to Ω.

We now consider clauses. We do not simply assert that each clause is sent to an inequality in Ω, as that is not the practice in many-valued logic programming. We must allow unit clauses, i.e., clauses of the form $\varphi \leftarrow$, to be assigned values other than 1. We do this as follows.

Definition 9. *Given a many-valued annotation-free logic program* Γ *over the language* $\mathcal{L}$*, a* valuation v *of* Γ *in a preorder* Ω *with finite meets is an assignment to each unit clause* $\varphi \leftarrow$ *of* Γ *of an element* $v(\varphi \leftarrow)$ *of* Ω*.*

The notion of a valuation is often used in many-valued logic programming to describe a map from the elements of the Herbrand base to Ω. In our setting, the latter map would be redundant. Using Definition 9, we can interpret clauses directly, as follows.

Definition 10. *Given an annotation-free logic program Γ over the language $\mathcal{L}$, and a valuation v of Γ, a* ground interpretation *of Γ* with respect to the valuation v *in a preorder Ω with finite meets is an interpretation $|\ |: p_{\mathcal{L}}(1) \to \Omega$ of $\mathcal{L}$ such that for each clause in Γ of the form $\varphi \leftarrow \overline{\varphi}$, with $\overline{\varphi}$ non-empty, and each ground substitution $[g]$,*

$$|\varphi_1[g]| \wedge \ldots \wedge |\varphi_n[g]| \leq |\varphi[g]|,$$

and, for each unit clause $\varphi \leftarrow$ and ground substitution g, $v(\varphi \leftarrow) \leq |\varphi[g]|$.

Due to its inductive nature, this definition corresponds to the notion of the semantic operator (and its iterations) for many-valued logic programs; that is, the ground interpretation of a program is computed stepwise, starting with formulae which have received their valuation and then computing values for the rest of the formulae using the given data.

Example 7. If we fix $[0,1]$ to be the chosen preorder, then a valuation for the logic program GC from Example 1 can be given as follows.
$v(\text{connected}(x,x) \leftarrow) = 1,\ v(\text{edge}(a,b) \leftarrow) = 0.75,\ v(\text{edge}(b,c) \leftarrow) = 0.25$

The minimal ground interpretation would be given by
$|\text{connected}(a,a)| = 1,\ |\text{connected}(b,b)| = 1,\ |\text{connected}(c,c)| = 1;$
$\min(|\text{edge}(a,b)| = 0.75, |\text{connected}(b,b)| = 1) \ \leq\ |\text{connected}(a,b)| = 0.75,$
$\min(|\text{edge}(b,c)| = 0.25, |\text{connected}(c,c)| = 1) \ \leq\ |\text{connected}(b,c)| = 0.25,$
$\min(|\text{edge}(a,b)| = 0.75, |\text{connected}(b,c)| = 0.25) \ \leq\ |\text{connected}(a,c)| = 0.25,$
$|\text{edge}(a,b)| = 0.75,\ |\text{edge}(b,c)| = 0.25$

There is a standard way of defining a minimal model for many-valued logic programs, described, for example, in [7,4]. We can emulate this in our own terms by defining an ordering on a set of all the *ground interpretations* as follows. Let $|\ |_1$ and $|\ |_2$ be ground interpretations with respect to a valuation v for a logic program Γ over the language $\mathcal{L}$. Then we say that $|\ |_1 \leq |\ |_2$ if $|\varphi[g]|_1 \leq |\varphi[g]|_2$ for every ground substitution of every formula φ in Γ. The set of all ground interpretations forms a preorder M with objects the ground interpretations and arrows given by $\leq$ defined as above. We define the *ground model* of an annotation-free logic program Γ to be the least element of M.

One needs to be careful in regard to the ground models as the following examples illustrate.

Example 8. Consider a logic program of the form $p(a) \leftarrow$, $p(x) \leftarrow q(x)$, $q(a) \leftarrow$, with valuation v in $[0,1]$ given by $v(p(a) \leftarrow) = 0.3$; $v(q(a) \leftarrow) = 0.7$.

By Definition 10, in any ground interpretation, $0.3 \leq |p(a)|$, $0.7 \leq |q(a)|$, and also $|q(a)| \leq |p(a)|$. Thus, $0.7 \leq |p(a)|$ in any ground interpretation. So, there is a one-step proof that $0.3 \leq |p(a)|$ and a two-step proof that $0.7 \leq |p(a)|$. This situation evidently can be extended to logic programs involving proofs of indefinite length, so needs to be taken seriously when giving SLD-resolution, in particular in determining the ground model.

Example 9. Consider the logic program of Example 8 with valuation in $[0,1] \times [0,1]$ given by $v(p(a) \leftarrow) = (0, 0.5)$; $v(q(a) \leftarrow) = (0.5, 0)$. Then, in any ground interpretation, $(0.5, 0) \leq |p(a)|$ and $(0, 0.5) \leq |p(a)|$, so $(0.5, 0.5) \leq |p(a)|$, but there is no computation that shows this directly. This will lead us to requiring finite joins in Ω in Section 6. Variants of this example exist in Kleene's logics and logics which generalise Kleene's logics, [6,7].

5 Fibrational Many-Valued Semantics

The fibrational semantics will provide us with non-ground interpretations for logic programs. In Theorem 1 we relate ground and fibrational semantics.

Let $\mathcal{C}$ be a small category and $\mathcal{D}$ have all products, and let 1 be a terminal object of $\mathcal{C}$. The diagonal functor $\Delta : \mathcal{D} \to \mathcal{D}^{\mathcal{C}^{\mathrm{op}}}$ has a right adjoint given by sending $F \in \mathcal{D}^{\mathcal{C}^{\mathrm{op}}}$ to $F(1)$. I.e., a right adjoint to the diagonal is given by evaluation at 1, and we will denote the right adjoint by $ev_1 : \mathcal{D}^{\mathcal{C}^{\mathrm{op}}} \to \mathcal{D}$.

Proposition 1. *The functor $ev_1 : \mathcal{D}^{\mathcal{C}^{op}} \to \mathcal{D}$ has a right adjoint $R : \mathcal{D} \to \mathcal{D}^{\mathcal{C}^{op}}$, given by $R(D)(C) = D^{\mathcal{C}(1,C)}$, for each $D \in \mathcal{D}$ and each $C \in \mathcal{C}^{op}$.*

Corollary 1. *The functor $ev_1 : \mathtt{FP}_s^{C_{\mathcal{T},\mathcal{F}}} \to \mathtt{FP}_s$ has a right adjoint given by $R(\Omega) = \Omega^{C_{\mathcal{T},\mathcal{F}}(1,-)}$.*

Recall that in Section 4, we studied maps of the form $p_{\mathcal{L}}(1) \to \Omega$ in $\mathtt{FP}_s$. By Corollary 1, they are equivalent to natural transformation $p_{\mathcal{L}} \to \Omega^{C_{\mathcal{T},\mathcal{F}}(1,-)}$. So, consider a natural transformation $\psi : p_{\mathcal{L}} \to \Omega^{C_{\mathcal{T},\mathcal{F}}(1,-)}$. This is equivalent to giving, for each $\overline{T}$ and each ground term $\overline{t}$ of sort $\overline{T}$, a finite meet preserving function $|\ | : p_{\mathcal{L}}(1) \to \Omega$ natural in $\overline{T}$.

Since Ω is a preorder with finite meets, $\Omega^{C_{\mathcal{T},\mathcal{F}}(1,\overline{T})}$ has a preorder structure with finite meet given pointwise. We use that fact in our definition of fibrational interpretation, which by the above discussion will be equivalent to Definition 10.

Definition 11. *Given an annotation-free logic program Γ over the language $\mathcal{L}$, a* fibrational interpretation*, or* f-interpretation*, with respect to the valuation v of Γ in Ω is given by an interpretation $\|\ \|$ of $\mathcal{L}$ in $\Omega^{C_{\mathcal{T},\mathcal{F}}(1,-)}$, such that:*

- *For each unit clause $\varphi \leftarrow$ in Γ, $v(\varphi \leftarrow) \leq \|\varphi\|$;*
- *For each clause in Γ of the form $\varphi \leftarrow \overline{\varphi}$, where $\overline{\varphi}$ is non-empty,*

$$\|\varphi_1\| \wedge \ldots \wedge \|\varphi_n\| \leq \|\varphi\|.$$

Theorem 1. *Given an annotation-free logic program Γ over the language $\mathcal{L}$, a preorder Ω, and a valuation v of Γ in Ω, to give an f-interpretation with respect to v is equivalent to giving a ground interpretation of Γ with respect to v.*

Proof. This follows from the adjointness of Corollary 1 and the definition of interpretation and valuation.

Example 10. We take the valuation of the program GC from Example 7. The minimal f-interpretation generated by the valuation is:
$\|\text{connected}(x,x)\| = 1$, $\|\text{edge}(a,b)\| = 0.75$, $\|\text{edge}(b,c)\| = 0.25$,
$\min(\|\text{edge}(x,z)\|, \|\text{connected}(z,y)\|) \leq \|\text{connected}(x,y)\|$.

The last line subsumes all the possible substitutions. Notably, ground substitutions agree with the ground interpretation for GC from Example 7.

Given a logic program Γ, we will call the least f-interpretation of Γ an *f-model* for Γ. It is the least element in the preorder of all f-interpretations of Γ, similarly to Section 4.

6 SLD-Resolution

Motivated by our fibrational semantics, we give a definition of the SLD-resolution for annotation-free logic programs. The idea is as follows. The syntax of annotation-free logic programs is exactly the same as that of conventional logic programs. So we can first do SLD-resolution for an annotation-free logic program qua conventional logic program which is expressible in terms of fibrational semantics and is sound and complete with respect to fibrational semantics; [9]. Now we introduce valuations. Given a refutation tree, we consider the leaves. These amount to unit clauses, so have valuation. We then proceed in the backward direction from the leaves to the root of the refutation tree to generate a minimal value for the substituted goal. Note that the leaves are not necessarily ground, and hence fibrational rather than ground approach is appropriate.

We restrict the choice of Ω by requiring Ω to have all, not only finite, meets. The existence of all meets in Ω implies the existence of all joins. A delicate analysis allows us to restrict to finite joins in addition to finite meets. As Example 9 indicates, we need some such assumption in order to justify the existence of ground models and f-models for annotation-free logic programs.

We start with a definition of SLD-resolution in terms of state transition machines. See also [9], where mgus were characterised as *pullbacks*. We will call $[s_1, s_2]$ an mgu of atomic formulae A and B with terms modelled by arrows u respectively v in $C_{\mathcal{T},\mathcal{F}}$, if $[s_1, s_2]$ is an mgu of u and v.

Definition 12. *Given an annotation-free logic program Γ in $\mathcal{L}$, the* state transition machine *M_Γ associated to Γ is the directed graph (N, E) defined as follows. N is the set of all formulae in $\mathcal{L}$. An edge with source $\varphi = \varphi_1 \times \ldots \times \varphi_n$ is a triple $(l, \rho, (s_1, s_2))$, where $l : H \leftarrow B$ is a clause in Γ, $\rho = \pi_i : \overline{\varphi} \to \varphi_i$ is the projection to φ_i, and (s_1, s_2) is an mgu for φ_i and H. The target of $(l, \rho, (s_1, s_2))$ is $\varphi_1[s_1, s_2] \times \ldots \times \varphi_{i-1}[s_1, s_2] \times B[s_1, s_2] \times \varphi_{i+1}[s_1, s_2] \times \ldots \times \varphi_n[s_1, s_2]$.*

Definition 13. *Given a logic program Γ and a goal G in $\mathcal{L}$, a* computation *in M_Γ with goal G is a directed path T in M_Γ starting at G, in particular, if the endpoint is a terminal 1 in some fibre of $p_\mathcal{L}$, then it is said to be a* successful computation *or* refutation. *Finally, if*

$$G = \overline{\varphi} \xrightarrow{(l_1,\rho_1,(s_1^1,s_2^1))} \overline{\varphi} \xrightarrow{(l_2,\rho_2,(s_1^2,s_2^2))} \cdots \xrightarrow{(l_{m-1},\rho_{m-1},(s_1^{m-1},s_2^{m-1}))} \overline{\varphi}$$

is a computation, $(s_1^1, s_2^1), (s_1^2, s_2^2), \ldots, (s_1^{m-1}, s_2^{m-1})$ *is defined to be its* answer.

The SLD-refutation is sound and complete with respect to the (two-valued) fibrational semantics, that is, the following theorem holds:

Theorem 2. *[9] Let Γ be a logic program in $\mathcal{L}$. Substitution $s : \overline{U} \to \overline{T}$ is the answer of a refutation in M_Γ with goal G of sort $\overline{T}$ if and only if there is an arrow $m : 1 \to G[s]$ in the fibre $p_\Gamma(\overline{U})$.*

Next we introduce a valuation into a mechanism of refutation to the annotation-free logic programs and give the inductive definition of a tree computing a value for the goal G as follows.

Definition 14. *Let M_Γ be the state transition machine associated to a logic program Γ and a goal G as in Definition 12. Let T be a directed path in M_Γ such that T performs a refutation for a formula G in $\mathcal{L}$ with the computed answer s, and let v be a valuation of Γ. A* computation of a value *for G is a directed path T^{op} starting at 1 and ending at $\|G[s]\|$ in Ω, such that the following holds:*

1. *Whenever there is an edge $(l, \rho, (s_1, s_2))$ from $\varphi_1[s_1, s_2] \times \ldots \times \varphi_{i-1}[s_1, s_2] \times \varphi_{i+1}[s_1, s_2] \times, \ldots, \times \varphi_n[s_1, s_2]$ to $\overline{\varphi} = \varphi_1 \times \ldots \times \varphi_i \times \ldots \times \varphi_n$, as described in Definition 12, with $\rho = \pi_i$ and l of the form $H \leftarrow$, then we use the valuation v of H and substitute φ_i in $\overline{\varphi}$ by $v(H)$.*
2. *For every edge $(l, \rho, (s_1, s_2))$ from $\varphi_1[s_1, s_2] \times \ldots \times \varphi_{i-1}[s_1, s_2] \times B_1[s_1, s_2], \ldots, B_k[s_1, s_2] \times \varphi_{i+1}[s_1, s_2] \times, \ldots, \times \varphi_n[s_1, s_2]$ to $\varphi = \varphi_1 \times \ldots \times \varphi_n$, with $\rho = \pi_i$ and l of the form $H \leftarrow B_1, \ldots B_k$, we use $v(B_1) \wedge \ldots \wedge v(B_k)$ to transform the node $\overline{\varphi}$ into $\varphi_1 \times \ldots \times \varphi_{i-1} \wedge (v(B_1) \wedge \ldots \wedge v(B_k)) \wedge \varphi_{i+1} \times \ldots \times \varphi_n$.*

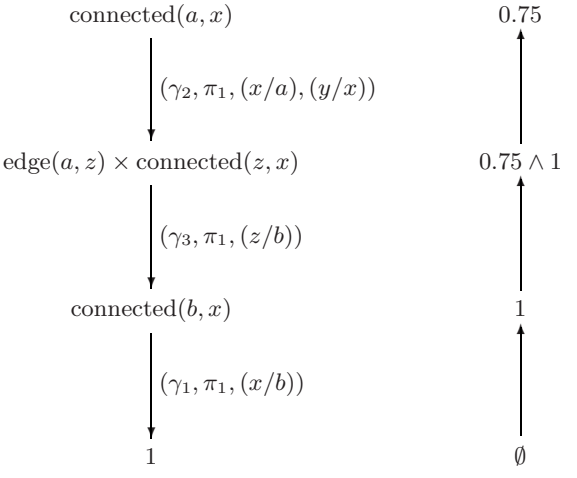
It is easy to see that for every such *computation of a value* for $G = \varphi'_1 \times \ldots \times \varphi'_m$, the endpont of T^{op} will be $\|G[s]\| = \bigwedge(v(B'_1[s])) \wedge \ldots \wedge \bigwedge(v(B'_m[s]))$, where each $\bigwedge(v(B'_j[s]))$ performs the value of the goal atom $\varphi'_j[s]$.

Definition 15. *Let Γ be an annotation-free logic program interpreted in a pre-order Ω with the least element 0. Let G be a goal in Γ. We say that $\omega \in \Omega$ is a* computed value *for G if one of the following conditions holds:*

- *There is a refutation for G with answer s and the algorithm of* computation of a value *outputs $\|G[s]\| = \omega$;*
- *There is no refutation for G and we put $\omega = 0$.*

Example 11. Consider the logic program GC from Example 1 and the goal G of the form (connected(a, x)). The leftmost tree T performs a refutation for G with the answer $s = (x/b)$. The rightmost tree T^{op} shows how the value for $G[s]$

is computed by the algorithm of computation of a value. We use the valuation v from Example 10.



Thus, the goal (connected$(a, x)[x/b]$) receives the computed value 0.75. Note that this agrees with the minimal ground interpretation of GC from Example 7 and f-model of GC from Example 10.

The algorithm of *computation of a value* for $G[s]$ is sound and complete with respect to both ground model and f-model of a logic program.

Theorem 3 (Soundness relative to fibrational semantics). *Let Γ be a logic program and G be a goal formula, such that there is a tree T in the state transition machine M_Γ and T performs refutation for $\{\Gamma \cup G\}$ with the computed answer s. Then the following holds. If the algorithm of* computation of a value *outputs the value ω for $G[s]$, then in the f-model of Γ, $\|G[s]\| \geq \omega$.*

Proof. We use Theorem 2; the rest of the proof proceeds by induction on the length of the tree $T \in M_\Gamma$.

Theorem 4 (Completeness relative to fibrational semantics). *If $\|G[s]\| = \omega$ is in the f-model of Γ, then there exists a finite set of trees $T_1, \ldots, T_n$ which compute the substitution s as answer, such that ω is the supremum of the computed values for $G[s]$ in $T_1, \ldots, T_n$.*

Proof. We use Theorem 2; then we proceed by induction on complexity of clauses interpreted by the ground model of Γ. Finite joins are required in order to account for cases such as Examples 8, 9. Only finite joins are needed as each valuation v only makes finitely many assignments. (Note that as we have assumed the existence of all meets in Ω, it follows that Ω also has finite joins.)

In practice, one needs to use the conventional algorithm of backtracking to compute all the values. Annotation-free logic programs can have infinitely long computations and infinitely long trees T in M_Γ, as in Example 12.

Example 12. The following logic program may have infinitely long refutations for the goal ($\leftarrow p(x)$): $q(x) \leftarrow$, $p(x) \leftarrow q(x)$, $q(x) \leftarrow p(x)$.

But the number of unit clauses in any logic program is finite, and so is the number of values assigned to them. This is why, refutations for annotation-free logic programs will always have finitely many computed values. In our example, the only possible computed value for $p(x)$ will be the value $v(q(x) \leftarrow)$.

We now show that traditional-style soundness and completeness of the SLD-resolution relative to the ground semantics can be obtained as a corollary of Theorems 1, 3, 4. We make use of Theorem 1 and use $|\ |$ instead of $\|\ \|$ when talking about interpretations for ground atoms.

In conventional logic programming [12], one speaks of the *success set* of a program Γ. That is the set of all ground atoms for which refutation exists. We cannot directly use that definition here because of the presence of non-trivial values. But, to give the success set of a conventional logic program is equivalent to giving function from $p_{\mathcal{L}}(1)$ to $\{0, 1\}$, satisfying a success condition. We could call that the success map corresponding to the success set, cf. [5]. So we generalise the success map as follows.

Definition 16. *Given an annotation-free logic program Γ over $\mathcal{L}$, a preorder Ω with meets, and a valuation v of Γ in Ω, the* success map *of Γ is the map $|\ | : p_{\mathcal{L}}(1) \to \Omega$ such that for each ground instance $\varphi[g]$ of a formula φ, $|\varphi[g]|$ is the supremum of all computed values ω of $\varphi[g]$.*

The soundness and completeness of the algorithm of *computation of a value* relative to the ground model of a given program can now be stated as follows.

Corollary 2 (Soundness and completeness relative to ground semantics). *Let Γ be a many-valued annotation-free logic program. The success map of Γ is equal to its ground model.*

7 Conclusions and Further Work

We have given ground and fibrational semantics to many-valued logic programming. We have proved theorems showing the exact relationship between the two kinds of semantics. This gave theoretical justification of the appropriateness of the fibrational (non-ground) approach to logic programming semantics. Fibrational semantics easily relates to existing resolution procedures [9] and gives rise to novel proof search algorithms. In particular, we have developed the novel algorithm of SLD-resolution for annotation-free many-valued logic programs. We proved that this algorithm is sound and complete relative to the fibrational semantics and showed that soundness and completeness of the algorithm relative to ground semantics can be obtained as a corollary of that.

Related Work. Comparing with other kinds of many-valued resolution algorithms [3,16], the algorithm we have described turns out to be the ideal compromise between expressiveness and efficiency. Unlike [16], we do not impose

syntactical restrictions on the shape and groundness of clauses and goals; and instead allow any first-order definite logic program to be processed. E.g., programs as in Example 12 would not be allowed in the setting of [16]. So, fibrational approach gives a clear advantage of expressiveness.

The algorithm of [3] is not restricting the syntax of logic programs, but it is very complex and in general non-terminating. In [3] one has to go through 5 consecutive stages of building a forest the trees of which would present all possible branches of refutation. Our algorithm avoids this by using conventional method of backtracking to find all the possible values and substitutions.

It is remarkable that both [3] and [16] use ground semantics in order to prove soundness and completeness of the algorithms. And this adds complications. Thus, proofs of soundness and completeness of the resolution in [3] exclude the non-terminating cases like the one in Example 12. So, it would seem that the shift towards non-ground, fibrational approach to resolution simplifies the algorithm as well as makes proofs of soundness and completeness easier.

Further work may involve extensions of the fibrational semantics to other types of many-valued logic programs: implication-based and annotated (signed).

We intentionally analysed only the simplest type of logic programs, that is, definite programs which allow only conjunctions in clause bodies. One can adapt existing categorical interpretations of other connectives [11] to the setting.

We also hope that the result relating ground and fibrational semantics will open new horizons for the structural characterisation of other types of non-classical logic programs (such as (e.g.) multimodal, non-monotonic) whose declarative semantics depends on truth assignments.

References

1. Baaz, M., Fermuller, C.G., Sazler, G.: Automated deduction for many-valued logics. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 2, pp. 1355–1402. Elsevier, Amsterdam (2001)
2. Barr, M., Wells, C.: Category Theory for Computing Science. Prentice-Hall, Englewood Cliffs (1990)
3. Damásio, C., Medina, M., Ojeda-Aciego, J.: A tabulation procedure for first-order residuated logic programs. In: Proc. IPMU 2006 (2006)
4. Damásio, C.V., Pereira, L.M.: Sorted monotonic logic programs and their embeddings. In: Proc. IPMU 2004, pp. 807–814 (2004)
5. Finkelstein, S.E., Freyd, P., Lipton, J.: Logic programming in tau categories. In: Pacholski, L., Tiuryn, J. (eds.) CSL 1994. LNCS, vol. 933. Springer, Heidelberg (1995)
6. Fitting, M.: A Kripke/Kleene semantics for logic programs. J. of logic programming 2, 295–312 (1985)
7. Fitting, M.: Fixpoint semantics for logic programming — a survey. TCS 278(1-2), 25–51 (2002)
8. Kifer, M., Subrahmanian, V.S.: Theory of generalized annotated logic programming and its applications. J. of logic programming 12, 335–367 (1991)
9. Kinoshita, Y., Power, A.J.: A fibrational semantics for logic programs. In: Herre, H., Dyckhoff, R., Schroeder-Heister, P. (eds.) ELP 1996. LNCS, vol. 1050. Springer, Heidelberg (1996)

10. Komendantskaya, E., Power, J.: Fibrational semantics for many-valued logic programs. Tech. Report N 00295027, INRIA (2008), http://hal.inria.fr/inria-00295027/en/
11. Lambek, J., Scott, P.: Higher Order Categorical Logic. Cambridge University Press, Cambridge (1986)
12. Lloyd, J.: Foundations of Logic Programming, 2nd edn. Springer, Heidelberg (1987)
13. Lu, J.J., Murray, N.V., Rosenthal, E.: Deduction and search strategies for regular multiple-valued logics. J. of Multiple-valued logic and soft computing 11, 375–406 (2005)
14. MacLane, S.: Categories for the working mathematician. Springer, Berlin (1971)
15. Power, A.J., Sterling, L.: A notion of a map between logic programs. In: Logic Programming, Proc. 7th Int. Conf., pp. 390–404. MIT Press, Cambridge (1990)
16. Straccia, U.: A top-down query answering procedure for normal logic programs under the any-world assumption. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 115–127. Springer, Heidelberg (2007)
17. van Emden, M.: Quantitative deduction and fixpoint theory. J. Logic Programming 3, 37–53 (1986)

Confluence Operators

Sébastien Konieczny[1] and Ramón Pino Pérez[2]

[1] CRIL - CNRS
Faculté des Sciences, Université d'Artois, Lens, France
konieczny@cril.fr
[2] Departamento de Matemáticas
Facultad de ciencias, Universidad de Los Andes, Mérida, Venezuela
pino@ula.ve

Abstract. In the logic based framework of knowledge representation and reasoning many operators have been defined in order to capture different kinds of change: revision, update, merging and many others. There are close links between revision, update, and merging. Merging operators can be considered as extensions of revision operators to multiple belief bases. And update operators can be considered as pointwise revision, looking at each model of the base, instead of taking the base as a whole. Thus, a natural question is the following one: Are there natural operators that are pointwise merging, just as update are pointwise revision? The goal of this work is to give a positive answer to this question. In order to do that, we introduce a new class of operators: the confluence operators. These new operators can be useful in modelling negotiation processes.

1 Introduction

Belief change theory has produced a lot of different operators that models the different ways the beliefs of one (or some) agent(s) evolve over time. Among these operators, one can quote revision [1,5,10,6], update [9,8], merging [19,14], abduction [16], extrapolation [4], etc.

In this paper we will focus on revision, update and merging. Let us first briefly describe these operators informally:

Revision. Belief revision is the process of accomodating a new piece of evidence that is more reliable than the current beliefs of the agent. In belief revision the world is static, it is the beliefs of the agents that evolve.

Update. In belief update the new piece of evidence denotes a change in the world. The world is dynamic, and these (observed) changes modify the beliefs of the agent.

Merging. Belief merging is the process of defining the beliefs of a group of agents. So the question is: Given a set of agents that have their own beliefs, what can be considered as the beliefs of the group?

Apart from these intuitive differences between these operators, there are also close links between them. This is particularly clear when looking at the technical definitions. There are close relationship between revision [1,5,10] and KM update operators [9]. The first ones looking at the beliefs of the agents globally, the second ones looking at them locally (this sentence will be made formally clear later in the paper)[1]. There is

[1] See [8,4,15] for more discussions on update and its links with revision.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 272–284, 2008.

also a close connection between revision and merging operators. In fact revision operators can be seen as particular cases of merging operators. From these two facts a very natural question arises: What is the family of operators that are a generalization of update operators in the same way merging operators generalize revision operators? Or, equivalently, what are the operators that can be considered as pointwise merging, just as KM update operators can be considered as pointwise belief revision. This can be outlined in the figure below. The aim of this paper is to introduce and study the operators corresponding to the question mark. We will call these new operators confluence operators.

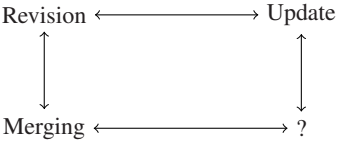


Fig. 1. Revision - Update - Merging - Confluence

These new operators are more cautious than merging operators. This suggest that they can be used to define negotiation operators (see [2, 20, 18, 17, 12]), or as a first step of a negotiation process, in order to find all the possible negotiation results.

In order to illustrate the need for these new operators and also the difference of behaviour between merging and confluence we present the following small example.

Example 1. Mary and Peter are planning to buy a car. Mary does not like a German car nor an expensive car. She likes small cars. Peter hesitates between a German, expensive but small car or a car which is not German, nor expensive and is a big car. Taking three propositional variables `German_car`, `Expensive_car` and `Small_car` in this order, Mary's desires are represented by $mod(A) = \{001\}$ and Peter's desires by $mod(B) = \{111, 000\}$. Most of the merging operators[2] give as solution (in semantical terms) the set $\{001, 000\}$. That is the same solution obtained when we suppose that Peter's desires are only a car which is not German nor expensive but a big car ($mod(B') = \{000\}$). The confluence operators will take into account the disjunctive nature of Peter's desires in a better manner and they will incorporate also the interpretations that are a trade-off between 001 and 111. For instance, the worlds 011 and 101 will be also in the solution if one use the confluence operator $\diamond^{d_H, Gmax}$ (defined in Section 7).

This kind of operators is particularly adequate when the base describes a situation that is not perfectly known, or that can evolve in the future. For instance Peter's desires can either be imperfectly known (he wants one of the two situations but we do not know which one), or can evolve in the future (he will choose later between the two situations). In these situations the solutions proposed by confluence operators will be more adequate than the one proposed by merging operators. The solutions proposed by the confluence operators can be seen as all possible agreements in a negotiation process.

[2] Such as $\triangle^{d_H, \Sigma}$ and $\triangle^{d_H, Gmax}$ [14].

In the next section we will give the required definitions and notations. In Section 3 we will recall the postulates and representation theorems for revision, update, and merging, and state the links between these operators. In Section 4 we define confluence operators. We provide a representation theorem for these operators in Section 5. In Section 6 we study the links between confluence operators and update and merging. In Section 7 we give examples of confluence operators. And we conclude in Section 8.

2 Preliminaries

We consider a propositional language $\mathcal{L}$ defined from a finite set of propositional variables $\mathcal{P}$ and the standard connectives, including $\top$ and $\bot$.

An interpretation ω is a total function from $\mathcal{P}$ to $\{0,1\}$. The set of all interpretations is denoted by $\mathcal{W}$. An interpretation ω is a model of a formula $\phi \in \mathcal{L}$ if and only if it makes it true in the usual truth functional way. $mod(\varphi)$ denotes the set of models of the formula φ, i.e., $mod(\varphi) = \{\omega \in \mathcal{W} \mid \omega \models \varphi\}$. When M is a set of models we denote by φ_M a formula such that $mod(\varphi_M) = M$.

A *base* K is a finite set of propositional formulae. In order to simplify the notations, in this work we will identify the base K with the formula φ which is the conjunction of the formulae of K[3].

A *profile* Ψ is a non-empty multi-set (bag) of bases $\Psi = \{\varphi_1, \ldots, \varphi_n\}$ (hence different agents are allowed to exhibit identical bases), and represents a group of n agents.

We denote by $\bigwedge \Psi$ the conjunction of bases of $\Psi = \{\varphi_1, \ldots, \varphi_n\}$, i.e., $\bigwedge \Psi = \varphi_1 \wedge \ldots \wedge \varphi_n$. A profile Ψ is said to be consistent if and only if $\bigwedge \Psi$ is consistent. The multi-set union is denoted by $\sqcup$.

A formula φ is complete if it has only one model. A profile Ψ is complete if all the bases of Ψ are complete formulae.

If $\leq$ denotes a pre-order on $\mathcal{W}$ (i.e., a reflexive and transitive relation), then $<$ denotes the associated strict order defined by $\omega < \omega'$ if and only if $\omega \leq \omega'$ and $\omega' \not\leq \omega$, and $\simeq$ denotes the associated equivalence relation defined by $\omega \simeq \omega'$ if and only if $\omega \leq \omega'$ and $\omega' \leq \omega$. A pre-order is *total* if $\forall \omega, \omega' \in \mathcal{W}$, $\omega \leq \omega'$ or $\omega' \leq \omega$. A pre-order that is not total is called *partial*. Let $\leq$ be a pre-order on A, and $B \subseteq A$, then $\min(B, \leq) = \{b \in B \mid \nexists a \in B \; a < b\}$.

3 Revision, Update and Merging

Let us now recall in this section some background on revision, update and merging, and their representation theorems in terms of pre-orders on interpretations. This will allow us to give the relationships between these operators.

[3] Some approaches are sensitive to syntactical representation. In that case it is important to distinguish between K and the conjonction of its formulae (see e.g. [13]). But operators of this work are all syntax independant.

3.1 Revision

Definition 1 (Katsuno-Mendelzon [10]). *An operator $\circ$ is an AGM belief revision operator if it satisfies the following properties:*

(R1) $\varphi \circ \mu \vdash \mu$
(R2) *If* $\varphi \wedge \mu \nvdash \bot$ *then* $\varphi \circ \mu \equiv \varphi \wedge \mu$
(R3) *If* $\mu \nvdash \bot$ *then* $\varphi \circ \mu \nvdash \bot$
(R4) *If* $\varphi_1 \equiv \varphi_2$ *and* $\mu_1 \equiv \mu_2$ *then* $\varphi_1 \circ \mu_1 \equiv \varphi_2 \circ \mu_2$
(R5) $(\varphi \circ \mu) \wedge \phi \vdash \varphi \circ (\mu \wedge \phi)$
(R6) *If* $(\varphi \circ \mu) \wedge \phi \nvdash \bot$ *then* $\varphi \circ (\mu \wedge \phi) \vdash (\varphi \circ \mu) \wedge \phi$

When one works with a finite propositional language the previous postulates, proposed by Katsuno and Mendelzon, are equivalent to AGM ones [1,5]. In [10] Katsuno and Mendelzon give also a representation theorem for revision operators, showing that each revision operator corresponds to a faithful assignment, that associates to each base a plausibility preorder on interpretations (this idea can be traced back to Grove systems of spheres [7]).

Definition 2. *A faithful assignment is a function mapping each base φ to a pre-order $\leq_\varphi$ over interpretations such that:*

1. *If* $\omega \models \varphi$ *and* $\omega' \models \varphi$, *then* $\omega \simeq_\varphi \omega'$
2. *If* $\omega \models \varphi$ *and* $\omega' \not\models \varphi$, *then* $\omega <_\varphi \omega'$
3. *If* $\varphi \equiv \varphi'$, *then* $\leq_\varphi = \leq_{\varphi'}$

Theorem 1 (Katsuno-Mendelzon [10]). *An operator $\circ$ is a revision operator (ie. it satisfies (R1)-(R6)) if and only if there exists a faithful assignment that maps each base φ to a total pre-order $\leq_\varphi$ such that*

$$mod(\varphi \circ \mu) = \min(mod(\mu), \leq_\varphi).$$

This representation theorem is important because it provides a way to easily define revision operators by defining faithful assignments. But also because their are similar such theorems for update and merging (we will also show a similar result for confluence), and that these representations in term of assignments allow to more easily find links between these operators.

3.2 Update

Definition 3 (Katsuno-Mendelzon [9,11]). *An operator $\diamond$ is a (partial) update operator if it satisfies the properties* **(U1)-(U8)**. *It is a total update operator if it satisfies the properties* **(U1)-(U5), (U8), (U9)**.

(U1) $\varphi \diamond \mu \vdash \mu$
(U2) *If* $\varphi \vdash \mu$, *then* $\varphi \diamond \mu \equiv \varphi$
(U3) *If* $\varphi \nvdash \bot$ *and* $\mu \nvdash \bot$ *then* $\varphi \diamond \mu \nvdash \bot$
(U4) *If* $\varphi_1 \equiv \varphi_2$ *and* $\mu_1 \equiv \mu_2$ *then* $\varphi_1 \diamond \mu_1 \equiv \varphi_2 \diamond \mu_2$
(U5) $(\varphi \diamond \mu) \wedge \phi \vdash \varphi \diamond (\mu \wedge \phi)$

(U6) *If* $\varphi \diamond \mu_1 \vdash \mu_2$ *and* $\varphi \diamond \mu_2 \vdash \mu_1$, *then* $\varphi \diamond \mu_1 \equiv \varphi \diamond \mu_2$
(U7) *If* φ *is a complete formula, then* $(\varphi \diamond \mu_1) \wedge (\varphi \diamond \mu_2) \vdash \varphi \diamond (\mu_1 \vee \mu_2)$
(U8) $(\varphi_1 \vee \varphi_2) \diamond \mu \equiv (\varphi_1 \diamond \mu) \vee (\varphi_2 \diamond \mu)$
(U9) *If* φ *is a complete formula and* $(\varphi \diamond \mu) \wedge \phi \nvdash \bot$, *then* $\varphi \diamond (\mu \wedge \phi) \vdash (\varphi \diamond \mu) \wedge \phi$

As for revision, there is a representation theorem in terms of faithful assignment.

Definition 4. *A faithful assignment is a function mapping each interpretation* ω *to a pre-order* $\leq_\omega$ *over interpretations such that if* $\omega \neq \omega'$, *then* $\omega <_\omega \omega'$.

One can easily check that this faithful assignment on interpretations is just a special case of the faithful assignment on bases defined in the previous section on the complete base corresponding to the interpretation.

Katsuno and Mendelzon give two representation theorems for update operators. The first representation theorem corresponds to partial pre-orders.

Theorem 2 (Katsuno-Mendelzon [9,11]). *An update operator* $\diamond$ *satisfies (U1)-(U8) if and only if there exists a faithful assignment that maps each interpretation* ω *to a partial pre-order* $\leq_\omega$ *such that*

$$mod(\varphi \diamond \mu) = \bigcup_{\omega \models \varphi} \min(mod(\mu), \leq_{\varphi\{\omega\}})$$

And the second one corresponds to total pre-orders.

Theorem 3 (Katsuno-Mendelzon [9,11]). *An update operator* $\diamond$ *satisfies (U1)-(U5), (U8) and (U9) if and only if there exists a faithful assignment that maps each interpretation* ω *to a total pre-order* $\leq_\omega$ *such that*

$$mod(\varphi \diamond \mu) = \bigcup_{\omega \models \varphi} \min(mod(\mu), \leq_{\varphi\{\omega\}})$$

3.3 Merging

Definition 5 (Konieczny-Pino Pérez [14]). *An operator* $\triangle$ *mapping a pair* Ψ, μ *(profile, formula) into a formula denoted* $\triangle_\mu(\Psi)$ *is an IC merging operator if it satisfies the following properties:*

(IC0) $\triangle_\mu(\Psi) \vdash \mu$
(IC1) *If* μ *is consistent, then* $\triangle_\mu(\Psi)$ *is consistent*
(IC2) *If* $\bigwedge \Psi$ *is consistent with* μ, *then* $\triangle_\mu(\Psi) \equiv \bigwedge \Psi \wedge \mu$
(IC3) *If* $\Psi_1 \equiv \Psi_2$ *and* $\mu_1 \equiv \mu_2$, *then* $\triangle_{\mu_1}(\Psi_1) \equiv \triangle_{\mu_2}(\Psi_2)$
(IC4) *If* $\varphi_1 \vdash \mu$ *and* $\varphi_2 \vdash \mu$, *then* $\triangle_\mu(\{\varphi_1, \varphi_2\}) \wedge \varphi_1$ *is consistent if and only if* $\triangle_\mu(\{\varphi_1, \varphi_2\}) \wedge \varphi_2$ *is consistent*
(IC5) $\triangle_\mu(\Psi_1) \wedge \triangle_\mu(\Psi_2) \vdash \triangle_\mu(\Psi_1 \sqcup \Psi_2)$
(IC6) *If* $\triangle_\mu(\Psi_1) \wedge \triangle_\mu(\Psi_2)$ *is consistent, then* $\triangle_\mu(\Psi_1 \sqcup \Psi_2) \vdash \triangle_\mu(\Psi_1) \wedge \triangle_\mu(\Psi_2)$
(IC7) $\triangle_{\mu_1}(\Psi) \wedge \mu_2 \vdash \triangle_{\mu_1 \wedge \mu_2}(\Psi)$
(IC8) *If* $\triangle_{\mu_1}(\Psi) \wedge \mu_2$ *is consistent, then* $\triangle_{\mu_1 \wedge \mu_2}(\Psi) \vdash \triangle_{\mu_1}(\Psi)$

There is also a representation theorem for merging operators in terms of pre-orders on interpretations [14].

Definition 6. *A* syncretic assignment *is a function mapping each profile Ψ to a total pre-order $\leq_\Psi$ over interpretations such that:*

1. *If $\omega \models \Psi$ and $\omega' \models \Psi$, then $\omega \simeq_\Psi \omega'$*
2. *If $\omega \models \Psi$ and $\omega' \not\models \Psi$, then $\omega <_\Psi \omega'$*
3. *If $\Psi_1 \equiv \Psi_2$, then $\leq_{\Psi_1} = \leq_{\Psi_2}$*
4. *$\forall \omega \models \varphi \; \exists \omega' \models \varphi' \; \omega' \leq_{\{\varphi\} \sqcup \{\varphi\}'} \omega$*
5. *If $\omega \leq_{\Psi_1} \omega'$ and $\omega \leq_{\Psi_2} \omega'$, then $\omega \leq_{\Psi_1 \sqcup \Psi_2} \omega'$*
6. *If $\omega <_{\Psi_1} \omega'$ and $\omega \leq_{\Psi_2} \omega'$, then $\omega <_{\Psi_1 \sqcup \Psi_2} \omega'$*

Theorem 4 (Konieczny-Pino Pérez [14]). *An operator $\triangle$ is an IC merging operator if and only if there exists a syncretic assignment that maps each profile Ψ to a total pre-order $\leq_\Psi$ such that*

$$mod(\triangle_\mu(\Psi)) = \min(mod(\mu), \leq_\Psi)$$

3.4 Revision vs. Update

Intuitively revision operators bring a minimal change to the base by selecting the most plausible models among the models of the new information. Whereas update operators bring a minimal change to each possible world (model) of the base in order to take into account the change described by the new infomation whatever the possible world. So, if we look closely to the two representation theorems (propositions 1, 2 and 3), we easily find the following result:

Theorem 5. *If $\circ$ is a revision operator (i.e. it satisfies (R1)-(R6)), then the operator $\diamond$ defined by:*

$$\varphi \diamond \mu = \bigvee_{\omega \models \varphi} \varphi_{\{\omega\}} \circ \mu$$

is an update operator that satisfies (U1)-(U9).

Moreover, for each update operator $\diamond$, there exists a revision operator $\circ$ such that the previous equation holds.

As explained above this proposition states that update can be viewed as a kind of pointwise revision.

3.5 Revision vs. Merging

Intuitively revision operators select in a formula (the new evidence) the closest information to a ground information (the old base). And, identically, IC merging operators select in a formula (the integrity constraints) the closest information to a ground information (a profile of bases).

So following this idea it is easy to make a correspondence between IC merging operators and belief revision operators [14]:

Theorem 6 (Konieczny-Pino Pérez [14]). *If $\triangle$ is an IC merging operator (it satisfies (IC0-IC8)), then the operator $\circ$, defined as $\varphi \circ \mu = \triangle_\mu(\varphi)$, is an AGM revision operator (it satisfies (R1-R6)).*

See [14] for more links between belief revision and merging.

4 Confluence Operators

So now that we have made clear the connections sketched in figure 1 between revision, update and merging, let us turn now to the definition of confluence operators, that aim to be a pointwise merging, similarly as update is a pointwise revision, as explained in Section 3.4. Let us first define p-consistency for profiles.

Definition 7. *A profile* $\Psi = \{\varphi_1, \dots \varphi_n\}$ *is* p-consistent *if all its bases are consistent, i.e* $\forall \varphi_i \in \Psi$, φ_i *is consistent.*

Note that p-consistency is much weaker than consistency, the former just asks that all the bases of the profile are consistent, while the later asks that the conjunction of all the bases is consistent.

Definition 8. *An operator* $\diamond$ *is a confluence operator if it satisfies the following properties:*

(UC0) $\diamond_\mu(\Psi) \vdash \mu$
(UC1) *If* μ *is consistent and* Ψ *is p-consistent, then* $\diamond_\mu(\Psi)$ *is consistent*
(UC2) *If* Ψ *is complete,* Ψ *is consistent and* $\bigwedge \Psi \vdash \mu$, *then* $\diamond_\mu(\Psi) \equiv \bigwedge \Psi$
(UC3) *If* $\Psi_1 \equiv \Psi_2$ *and* $\mu_1 \equiv \mu_2$, *then* $\diamond_{\mu_1}(\Psi_1) \equiv \diamond_{\mu_2}(\Psi_2)$
(UC4) *If* φ_1 *and* φ_2 *are complete formulae and* $\varphi_1 \vdash \mu$, $\varphi_2 \vdash \mu$, *then* $\diamond_\mu(\{\varphi_1, \varphi_2\}) \wedge \varphi_1$ *is consistent if and only* $\diamond_\mu(\{\varphi_1, \varphi_2\}) \wedge \varphi_2$ *is consistent*
(UC5) $\diamond_\mu(\Psi_1) \wedge \diamond_\mu(\Psi_2) \vdash \diamond_\mu(\Psi_1 \sqcup \Psi_2)$
(UC6) *If* Ψ_1 *and* Ψ_2 *are complete profiles and* $\diamond_\mu(\Psi_1) \wedge \diamond_\mu(\Psi_2)$ *is consistent, then* $\diamond_\mu(\Psi_1 \sqcup \Psi_2) \vdash \diamond_\mu(\Psi_1) \wedge \diamond_\mu(\Psi_2)$
(UC7) $\diamond_{\mu_1}(\Psi) \wedge \mu_2 \vdash \diamond_{\mu_1 \wedge \mu_2}(\Psi)$
(UC8) *If* Ψ *is a complete profile and if* $\diamond_{\mu_1}(\Psi) \wedge \mu_2$ *is consistent, then* $\diamond_{\mu_1 \wedge \mu_2}(\Psi) \vdash \diamond_{\mu_1}(\Psi) \wedge \mu_2$
(UC9) $\diamond_\mu(\Psi \sqcup \{\varphi \vee \varphi'\}) \equiv \diamond_\mu(\Psi \sqcup \{\varphi\}) \vee \diamond_\mu(\Psi \sqcup \{\varphi'\})$

Some of the (UC) postulates are exactly the same as (IC) ones, just like some (U) postulates for update are exactly the same as (R) ones for revision.

In fact, (UC0), (UC3), (UC5) and (UC7) are exactly the same as the corresponding (IC) postulates. So the specificity of confluence operators lies in postulates (UC1), (UC2), (UC6), (UC8) and (UC9). (UC2), (UC4), (UC6) and (UC8) are close to the corresponding (IC) postulates, but hold for complete profiles only. The present formulation of (UC2) is quite similar to formulation of (U2) for update. Note that in the case of a complete profile the hypothesis of (UC2) is equivalent to ask coherence with the constraints, *i.e.* the hypothesis of (IC2). Postulates (UC8) and (UC9) are the main difference with merging postulates, and correspond also to the main difference between revision and KM update operators. (UC9) is the most important postulate, that defines confluence operators as pointwise agregation, just like (U8) defines update operators as pointwise revision. This will be expressed more formally in the next Section (Lemma 1).

5 Representation Theorem for Confluence Operators

In order to state the representation theorem for confluence operators, we first have to be able to "localize" the problem. For update this is done by looking to each model of the base, instead of looking at the base (set of models) as a whole. So for "localizing" the aggregation process, we have to find what is the local view of a profile. That is what we call a state.

Definition 9. *A multi-set of interpretations will be called a* state. *We use the letter* e, *possibly with subscripts, for denoting states. If* $\Psi = \{\varphi_1, \ldots, \varphi_n\}$ *is a profile and* $e = \{\omega_1, \ldots, \omega_n\}$ *is a state such that* $\omega_i \models \varphi_i$ *for each* i, *we say that* e *is a state of the profile* Ψ, *or that the state* e *models the profile* Ψ, *that will be denoted by* $e \models \Psi$. *If* $e = \{\omega_1, \ldots, \omega_n\}$ *is a state, we define the profile* Ψ_e *by putting* $\Psi_e = \{\varphi_{\{\omega_1\}}, \ldots, \varphi_{\{\omega_n\}}\}$.

State is an interesting notion. If we consider each base as the current point of view (goals) of the corresponding agent (that can be possibly strengthened in the future) then states are all possible negotiation starting points.

States are the points of interest for confluence operators (like interpretations are for update), as stated in the following Lemma:

Lemma 1. *If* $\diamond$ *satisfies (UC3) and (UC9) then* $\diamond$ *satisfies the following*

$$\diamond_\mu(\Psi) \equiv \bigvee_{e \models \Psi} \diamond_\mu(\Psi_e)$$

Defining profile entailment by putting $\Psi \vdash \Psi'$ iff every state of Ψ is a state of Ψ', the previous Lemma has as a corollary the following:

Corollary 1. *If* $\diamond$ *is a confluence operator then it is monotonic in the profiles, that means that if* $\Psi \vdash \Psi'$ *then* $\diamond_\mu(\Psi) \vdash \diamond_\mu(\Psi')$

This monotony property, that is not true in the case of merging operators, shows one of the big differences between merging and confluence operators. Remark that there is a corresponding monotony property for update.

Like revision's faithful assignments that have to be "localized" to interpretations for update, merging's syncretic assignments have to be localized to states for confluence.

Definition 10. *A* distributed assignment *is a function mapping each state* e *to a total pre-order* $\leq_e$ *over interpretations such that:*

1. $\omega <_{\{\omega,\ldots,\omega\}} \omega'$ *if* $\omega' \neq \omega$
2. $\omega \simeq_{\{\omega,\omega'\}} \omega'$
3. *If* $\omega \leq_{e_1} \omega'$ *and* $\omega \leq_{e_2} \omega'$, *then* $\omega \leq_{e_1 \sqcup e_2} \omega'$
4. *If* $\omega <_{e_1} \omega'$ *and* $\omega \leq_{e_2} \omega'$, *then* $\omega <_{e_1 \sqcup e_2} \omega'$

Now we can state the main result of this paper, that is the representation theorem for confluence operators.

Theorem 7. *An operator $\diamond$ is a confluence operator if and only if there exists a distributed assignment that maps each state e to a total pre-order $\leq_e$ such that*

$$mod(\diamond_\mu(\Psi)) = \bigcup_{e \models \Psi} \min(mod(\mu), \leq_e) \tag{1}$$

Unfortunately, we have to omit the proof for space reasons. Nevertheless, we indicate the most important ideas therein. As it is usual, the *if* condition is done by checking each property without any major difficulty. In order to verify the *only if* condition we have to define a distributed assigment. This is done in the following way: for each state e we define a total pre-order $\leq_e$ by putting $\forall \omega, \omega' \in \mathcal{W}\ \omega \leq_e \omega'$ if and only if $\omega \models \diamond_{\varphi_{\{\omega,\omega'\}}}(\Psi_e)$. Then, the main difficulties are to prove that this is indeed a distributed assigment and that the equation (1) holds. In particular, Lema 1 is very helpful for proving this last equation.

Note that this theorem is still true if we remove respectively the postulate (UC4) from the required postulates for confluence operators and the condition 2 from distributed assignments.

6 Confluence vs. Update and Merging

So now we are able to state the proposition that shows that update is a special case of confluence, just as revision is a special case of merging.

Theorem 8. *If $\diamond$ is a confluence operator (i.e. it satisfies (UC0-UC9)), then the operator $\diamond$, defined as $\varphi \diamond \mu = \diamond_\mu(\varphi)$, is an update operator (i.e. it satisfies (U1-U9)).*

Concerning merging operators, one can see easily that the restriction of a syncretic assignment to a complete profile is a distributed assignment. From that we obtain the following result (the one corresponding to Theorem 5):

Theorem 9. *If $\triangle$ is an IC merging operator (i.e. it satisfies (IC0-IC8)) then the operator $\diamond$ defined by*

$$\diamond_\mu(\Psi) = \bigvee_{e \models \Psi} \triangle_\mu(\Psi_e)$$

is a confluence operator (i.e. it satisfies (UC0-UC9)).

Moreover, for each confluence operator $\diamond$, there exists a merging operator $\triangle$ such that the previous equation holds.

It is interesting to note that this theorem shows that every merging operator can be used to define a confluence operator, and explains why we can consider confluence as a pointwise merging.

Unlike Theorem 5, the second part of the previous theorem doesn't follow straightforwardly from the representation theorems. We need to build a syncretic assignment extending the distributed assignment representing the confluence operator. In order to do that we can use the following construction: Each pre-order $\leq_e$ defines naturally a rank function r_e on natural numbers. Then we put

$$\omega \leq_\Psi \omega' \quad \text{if and only if} \quad \sum_{e \models \Psi} r_e(\omega) \leq \sum_{e \models \Psi} r_e(\omega')$$

As a corollary of the representation theorem we obtain the following

Corollary 2. *If $\diamond$ is a confluence operator then the following property holds:*

If $\bigwedge \Psi \vdash \mu$ and Ψ is consistent then $\bigwedge \Psi \wedge \mu \vdash \diamond_\mu(\Psi)$

But unlike merging operators, we don't have generally $\diamond_\mu(\Psi) \vdash \bigwedge \Psi \wedge \mu$.

Note that this "half of (IC2)" property is similar to the "half of (R2)" satisfied by update operators.

This corollary is interesting since it underlines an important difference between merging and confluence operators. If all the bases agree (i.e. if their conjunction is consistent), then a merging operator gives as result exactly the conjunction, whereas a confluence operator will give this conjunction plus additional results. This is useful if the bases do not represent interpretations that are considered equivalent by the agent, but uncertain information about the agent's current or future state of mind.

7 Example

In this section we will illustrate the behaviour of confluence operators on an example. We can define confluence operators very similarly to merging operators, by using a distance and an aggregation function.

Definition 11. *A* pseudo-distance *between interpretations is a total function $d : \mathcal{W} \times \mathcal{W} \mapsto \mathbb{R}^+$ s.t. for any ω, $\omega' \in \mathcal{W}$: $d(\omega, \omega') = d(\omega', \omega)$, and $d(\omega, \omega') = 0$ if and only if $\omega = \omega'$.*

A widely used distance between interpretations is the Dalal distance [3], denoted d_H, that is the Hamming distance between interpretations (the number of propositional atoms on which the two interpretations differ).

Definition 12. *An* aggregation function *f is a total function associating a nonnegative real number to every finite tuple of nonnegative real numbers s.t. for any $x_1, \ldots, x_n, x, y \in \mathbb{R}^+$:*

- *if $x \leq y$, then $f(x_1, \ldots, x, \ldots, x_n) \leq f(x_1, \ldots, y, \ldots, x_n)$* *(non-decreasingness)*
- *$f(x_1, \ldots, x_n) = 0$ if and only if $x_1 = \ldots = x_n = 0$* *(minimality)*
- *$f(x) = x$* *(identity)*

Sensible aggregation functions are for instance max, sum, or leximax ($Gmax$)[4] [14].

Definition 13 (distance-based confluence operators). *Let d be a pseudo-distance between interpretations and f be an aggregation function. The result $\diamond_\mu^{d,f}(\Psi)$ of the confluence of Ψ given the integrity constraints μ is defined by: $mod(\diamond_\mu^{d,f}(\Psi)) = \bigcup_{e \models \Psi} \min(mod(\mu), \leq_e)$, where the pre-order $\leq_e$ on $\mathcal{W}$ induced by e is defined by:*

[4] Leximax ($Gmax$) is usually defined using lexicographic sequences, but it can be easily represented by reals to fit the above definition (see e.g. [13]).

– $\omega \leq_e \omega'$ *if and only if* $d(\omega, e) \leq d(\omega', e)$, *where*
– $d(\omega, e) = f(d(\omega, \omega_1) \ldots, d(\omega, \omega_n))$ *with* $e = \{\omega_1, \ldots, \omega_n\}$.

It is easy to check that by using usual aggregation functions we obtain confluence operators.

Proposition 1. *Let* d *be any distance,* $\diamond_\mu^{d,\Sigma}(\Psi)$ *and* $\diamond_\mu^{d,Gmax}(\Psi)$ *are confluence operators (i.e. they satisfy (UC0)-(UC9)).*

Example 2. Let us consider a profile $\Psi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ and an integrity constraint μ defined on a propositional language built over four symbols, as follows: $mod(\mu) = \mathcal{W} \setminus \{0110, 1010, 1100, 1110\}$, $mod(\varphi_1) = mod(\varphi_2) = \{1111, 1110\}$, $mod(\varphi_3) = \{0000\}$, and $mod(\varphi_4) = \{1110, 0110\}$.

Table 1.

$\mathcal{W}$	1111	1110	0000	0110	e_1		e_2		e_3		e_4		e_5		e_6		$\diamond_\mu^{d,\Sigma}$	$\diamond_\mu^{d,\text{Gmax}}$
					Σ	Gmax	Σ	Gmax	Σ	Gmax	Σ	Gmax	Σ	Gmax	Σ	Gmax		
0000	4	3	0	2	11	4430	10	4420	10	4330	9	4320	9	3330	8	3320		
0001	3	4	1	3	11	4331	10	3331	12	4431	11	4331	13	4441	12	4431		
0010	3	2	1	1	9	3321	8	3311	8	3221	7	**3211**	**7**	**2221**	**6**	**2211**	×	×
0011	2	3	2	2	9	3222	8	**2222**	10	3322	9	3222	11	3332	10	3322		×
0100	3	2	1	1	9	3321	8	3311	8	**3221**	**7**	**3211**	**7**	**2221**	**6**	**2211**	×	×
0101	2	3	2	2	9	3222	8	**2222**	10	3322	9	3222	11	3332	10	3322		×
0110	2	1	2	0	7	2221	6	2220	6	2211	5	2210	5	2111	4	2110		
0111	1	2	3	1	7	**3211**	**6**	3111	8	**3221**	**7**	**3211**	9	3222	8	3221	×	×
1000	3	2	1	3	9	3321	10	3331	8	**3221**	9	3321	**7**	**2221**	8	3221	×	×
1001	2	3	2	4	9	3222	10	4222	10	3322	11	4322	11	3332	12	4332		
1010	2	1	2	2	7	2221	8	2222	6	2211	7	2221	5	2111	6	2211		
1011	1	2	3	3	7	**3211**	8	3311	8	**3221**	9	3321	9	3222	10	3322		×
1100	2	1	2	2	7	2221	8	2222	6	2211	7	2221	5	2111	6	2211		
1101	1	2	3	3	7	**3211**	8	3311	8	**3221**	9	3321	9	3222	10	3322		×
1110	1	0	3	1	5	3110	6	3111	4	3100	5	3110	3	3000	4	3100		
1111	0	1	4	2	**5**	4100	**6**	4200	**6**	4110	**7**	4210	**7**	4111	8	4211	×	

The computations are reported in Table 1. The shadowed lines correspond to the interpretations rejected by the integrity constraints. Thus the result has to be taken among the interpretations that are not shadowed. The states that model the profile are the following ones:
$e_1 = \{1111, 1111, 0000, 1110\}, e_2 = \{1111, 1111, 0000, 0110\}$,
$e_3 = \{1111, 1110, 0000, 1110\}, e_4 = \{1110, 1111, 0000, 0110\}$,
$e_5 = \{1110, 1110, 0000, 1110\}, e_6 = \{1110, 1110, 0000, 0110\}$.

For each state, the Table gives the distance between the interpretation and this state for the Σ aggregation function, and for the $Gmax$ one. So one can then look at the best interpretations for each state.

So for instance for $\diamond_\mu^{d,\Sigma}(\Psi)$, e_1 selects the interpretation 1111, e_2 selects 0111 and 1111, etc. So, taking the union of the interpretations selected by each state, gives $mod(\diamond_\mu^{d,\Sigma}(\Psi)) = \{0010, 0100, 0111, 1000, 1111\}$.

Similarly we obtain $mod(\diamond_\mu^{d,Gmax}(\Psi)) = \{0100, 0011, 0010, 0101, 0111, 1000, 1011, 1101\}$.

8 Conclusion

We have proposed in this paper a new family of change operators. Confluence operators are pointwise merging, just as update can be seen as a pointwise revision. We provide an axiomatic definition of this family, a representation theorem in terms of pre-orders on interpretations, and provide examples of these operators.

In this paper we define confluence operators as generalization to multiple bases of total update operators (i.e. which semantical counterpart are total pre-orders). A perspective of this work is to try to extend the result to partial update operators.

As Example 1 suggests, these operator can prove meaningful to aggregate the goals of a group of agents. They seem to be less adequate for aggregating beliefs, where the global minimization done by merging operators is more appropriate for finding the most plausible worlds. This distinction between goal and belief aggregation is a very interesting perspective, since, as far as we know, no such axiomatic distinction as been ever discussed.

Acknowledgements

The idea of this paper comes from discussions in the 2005 and 2007 Dagstuhl seminars (#5321 and #7351) *"Belief Change in Rational Agents"* . The authors would like to thank the Schloss Dagstuhl institution, and the participants of the seminars, especially Andreas Herzig for the initial question *"If merging can be seen as a generalization of revision, what is the generalization of update ?"*. Here is an answer !

The second author was partially supported by a research grant of the Mairie de Paris and by the project CDCHT-ULA N° C-1451-07-05-A. Part of this work was done when the second author was a visiting professor at CRIL (CNRS UMR 8188) from September to December 2007 and a visiting researcher at TSI Department of Telecom ParisTech (CNRS UMR 5141 LTCI) from January to April 2008. The second author thanks to CRIL and TSI Department for the excellent working conditions.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. Journal of Symbolic Logic 50, 510–530 (1985)
2. Booth, R.: Social contraction and belief negotiation. In: Proceedings of the Eighth Conference on Principles of Knowledge Representation and Reasoning (KR 2002), pp. 374–384 (2002)
3. Dalal, M.: Investigations into a theory of knowledge base revision: preliminary report. In: Proceedings of the American National Conference on Artificial Intelligence (AAAI 1988), pp. 475–479 (1988)
4. Dupin de Saint-Cyr, F., Lang, J.: Belief extrapolation (or how to reason about observations and unpredicted change). In: Proceedings of the Eighth Conference on Principles of Knowledge Representation and Reasoning (KR 2002), pp. 497–508 (2002)
5. Gärdenfors, P.: Knowledge in flux. MIT Press, Cambridge (1988)
6. Gärdenfors, P. (ed.): Belief Revision. Cambridge University Press, Cambridge (1992)

7. Grove, A.: Two modellings for theory change. Journal of Philosophical Logic 17(157-180) (1988)
8. Herzig, A., Rifi, O.: Update operations: a review. In: Proceedings of the Thirteenth European Conference on Artificial Intelligence (ECAI 1998), pp. 13–17 (1998)
9. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR 1991), pp. 387–394 (1991)
10. Katsuno, H., Mendelzon, A.O.: Propositional knowledge base revision and minimal change. Artificial Intelligence 52, 263–294 (1991)
11. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: Gärdenfors, P. (ed.) Belief Revision. Cambridge University Press, Cambridge (1992)
12. Konieczny, S.: Belief base merging as a game. Journal of Applied Non-Classical Logics 14(3), 275–294 (2004)
13. Konieczny, S., Lang, J., Marquis, P.: DA^2 merging operators. Artificial Intelligence 157(1-2), 49–79 (2004)
14. Konieczny, S., Pino Pérez, R.: Merging information under constraints: a logical framework. Journal of Logic and Computation 12(5), 773–808 (2002)
15. Lang, J.: Belief update revisited. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 2517–2522 (2007)
16. Lobo, J., Uzcátegui, C.: Abductive change operators. Fundamenta Informaticae 27(4), 385–411 (1996)
17. Meyer, T., Foo, N., Zhang, D., Kwok, R.: Logical foundations of negotiation: Outcome, concession and adaptation. In: Proceedings of the American National Conference on Artificial Intelligence (AAAI 2004), pp. 293–298 (2004)
18. Meyer, T., Foo, N., Zhang, D., Kwok, R.: Logical foundations of negotiation: Strategies and preferences. In: Proceedings of the Ninth Conference on Principles of Knowledge Representation and Reasoning (KR 2004), pp. 311–318 (2004)
19. Revesz, P.Z.: On the semantics of arbitration. International Journal of Algebra and Computation 7(2), 133–160 (1997)
20. Zhang, D., Foo, N., Meyer, T., Kwok, R.: Negotiation as mutual belief revision. In: Proceedings of the American National Conference on Artificial Intelligence (AAAI 2004), pp. 317–322 (2004)

A Game-Theoretic Measure of Argument Strength for Abstract Argumentation

Paul-Amaury Matt and Francesca Toni

Department of Computing,
Imperial College London
pmatt,f.toni@imperial.ac.uk

Abstract. Abstract argumentation (Dung 1995) is a theory of dialectic that allows us to formalise and study various notions of argument acceptability. We depart from this standard approach and formalise a measure of argument strength by applying the concept of value of a game, as defined in Game Theory (von Neumann 1928). The measure thus obtained satisfies a number of intuitively appealing properties that can be derived mathematically from the minimax theorem.

1 Introduction

Dialectic corresponds informally to the art or practise of logical discussion as employed in investigating the truth of a theory or an opinion. In classical philosophy, dialectic is controversy and consists in the exchange of arguments and counter-arguments respectively advocating propositions (theses) and counter-propositions (antitheses). Abstract argumentation (Dung 1995) can be seen as a modern theory of dialectic that allows us to model conflict between arguments and formalise various notions of argument acceptability. The adoption of arguments thus deemed dialectically acceptable combined with the rejection of unacceptable arguments constitutes a natural approach to deliberation.

This type of deliberation is however somewhat simplistic, as it classifies arguments into two categories only, *viz.*, acceptable and unacceptable arguments. Several works, e.g. (Krause *et al.* 1995, Jakobovits and Vermeir 1999, Besnard and Hunter 2001, Cayrol and Lagasquie-Schiex 2005) have considered and explored the possibility of discriminating between arguments using a larger number of categories or continuous numerical scales. The implicit common objective of such approaches is to eventually elaborate a theory of "careful deliberation" rooted in dialectic. We aim at following these works by assessing the strength of arguments on a scale of values ranging from 0 to 1 so as to finely compare and rank arguments in decreasing order of acceptability, identify the weakest arguments and better understand the influences that arguments have on each other in disputes. This fits well with recent interest in quantitative measures for the analysis of persuasion dialogues (Amgoud and Dupin de Saint-Cyr 2008, Budzyńska *et al.* 2008).

The most fundamental ideas used to formalise argument strength in this paper are essentially the same as those found in abstract argumentation theory: an

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 285–297, 2008.

argument may be called strong whenever the argument can be defended by one or several well-formed opinion(s) that properly withstand(s) external criticism. In order to assess the strength of a given argument in a dispute, we will essentially have to confront two fictitious persons, endorsing the roles of proponent and opponent of the argument. Situations of conflict between two persons such as this one can be rigorously analysed using the paradigm of Game Theory (von Neumann and Morgenstern 1944). We will thus introduce a special two-person game called *game of argumentation strategy* to confront the opinions of the proponent and opponent of an argument and assess its strength.

The remainder of the paper is organised as follows. In Section 2, we provide a short introduction to abstract argumentation theory. In Section 3, we set the exact rules of a game of argumentation strategy. In Section 4, we justify why the expected outcome of the game – also called game's value – may be adopted as strength value. We dedicate Section 5 to the mathematical study of this game-theoretic argument strength measure. We finally summarise the contribution of the paper and discuss related works in Section 6.

2 Abstract Argumentation

Arguments, opinions and the conflicts opposing opinions in a dispute can be represented in an elegant fashion using directed graphs whereby arguments appear as nodes and attacks between pairs of arguments appear as directed edges. Such graphs correspond to *abstract argumentation frameworks* and constitute the basis of abstract argumentation theory (Dung 1995). Formally,

Definition 1 (abstract argumentation framework). *An abstract argumentation framework is a pair* (Arg, att) *where* Arg *is a set of arguments and* $att \subseteq Arg \times Arg$ *is a binary relation between arguments.*

For example, a framework $F = (Arg, att)$ may consist of $Arg = \{a, b, c, d, e, f\}$ and $att = \{(a, b), (b, a), (b, c), (c, d), (e, c), (f, e)\}$. The corresponding directed graph is shown in Fig. 1.

The opinions held by the participants of a dispute can be simply represented by the sets of arguments they embrace. Thus, opinions formally correspond to sets of arguments $X \subseteq Arg$. Conflicts between opinions can be formalised in terms of attack between sets of arguments. We say that an opinion $X \subseteq Arg$

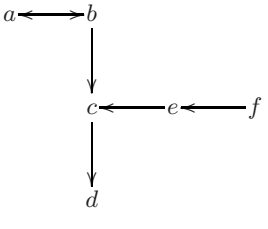


Fig. 1. A simple abstract argumentation framework

attacks the opinion $Y \subseteq Arg$ when there exists an attack $(x, y) \in X \times Y$ which originates from an argument $x \in X$ and is directed against an argument $y \in Y$. In the framework of Fig. 1, it holds for instance that $\{a, c, f\}$ attacks $\{b, e\}$, that $\{b, e\}$ attacks $\{c, d\}$ and that $\{c, d\}$ attacks itself.

The main purpose of argumentation theory is to identify which arguments and opinions are rationally "acceptable". To address this problem, several notions of *acceptability* have been put forward in the literature (Dung 1995, Bondarenko *et al.* 1997, Dung *et al.* 2006, 2007). In this paper, we will mostly deal with the notions of conflict-freeness, admissibility and stability.

Definition 2 (acceptability). *A set $X \subseteq Arg$ of arguments is said to be*

- *conflict-free if and only if X does not attack itself*
- *admissible if and only if X is conflict-free and attacks every argument that attacks X*
- *stable if and only if X is conflict-free and attacks every argument that is not an element of X*

Intuitively, conflict-freeness conveys the idea that well-formed opinions should be internally consistent. Admissibility is a stronger notion of acceptability according to which opinions should not only be conflict-free but also incorporate the counter-arguments necessary to resist (external) criticism. Finally stability is an even stronger notion of admissibility which requires all arguments not embraced by the opinion to be attacked.

In the example of Fig. 1, $\{a, c, f\}$ is conflict-free, admissible and stable, $\{b\}$ is conflict-free and admissible but not stable, $\{c, f\}$ is conflict-free but not admissible and $\{a, b, c\}$ is neither stable nor admissible nor conflict-free.

3 Games of Argumentation Strategy

In classical abstract argumentation, arguments are either acceptable or unacceptable, given a chosen notion of acceptability. This gives a rather coarse way to compare arguments. So, for example, for the framework given in Fig. 1, b and f are both equally admissible arguments. However, intuitively f can be deemed to be "stronger" than b, as it is not "weakened" by any attacking argument (whereas b is "weakened" by a). In general, in order to assess the strength of an argument, we will essentially weigh the opinions embracing that argument (opinions pro) against the possible criticisms that can be raised against them (opinions con). We will define, in Section 4, a notion of argument strength matching this intuition. This notion will be defined in terms of the value of a game of strategy (Borel 1921, von Neumann 1928, von Neumann and Morgenstern 1944) confronting two fictive players endorsing the roles of *proponent* and *opponent* of some argument of interest. In this section we will define this game.

Let us assume given and fixed an abstract argumentation framework (F, x) representing a dispute and denote by $x \in Arg$ the argument whose strength is to be measured. In the remainder of this section, we introduce the exact rules

of a game that is specific to the argument x and based on the structure of F. This game will be referred to as (F, x) *game of argumentation strategy*, or for convenience simply as (F, x) game.

In Game Theory, the choices available to the players are referred to as pure strategies. In the (F, x) game, strategies are sets of arguments $X \subseteq Arg$ and are interpreted as opinions. The proponent of argument x is required to embrace it, so we impose that x belongs to the set of arguments (strategy) played by the proponent. The opponent is however free to select any set of arguments to play the game.

Definition 3 (pure strategies). *The sets of pure strategies for the proponent and opponent players are* $\{P \,|\, P \subseteq Arg,\ x \in P\}$ *and* $\{O \,|\, O \subseteq Arg\}$ *respectively.*

Let (P, O) be an arbitrary pair of strategies chosen by the proponent and opponent respectively. A degree of acceptability of P with respect to O can be defined on the basis of the attacks directed from P to O and from O to P within the abstract argumentation framework F. Let us then denote for every set of arguments $A, B \subseteq Arg$ in the framework $F = (Arg, att)$

Notation 1 (set of attacks). $B_F^{\leftarrow A} = \{(a, b) \in A \times B \,|\, (a, b) \in att\}$

the set of attacks from A against B. According to this notation, $O_F^{\leftarrow P}$ represents the set of attacks from P against O and $P_F^{\leftarrow O}$ the set of attacks from O against P in F. In a dispute, it is better for the proponent of an argument to have more attacks against opponents to the argument and fewer attacks from them. To make sense dialectically, the degree of acceptability $\phi(P, O)$ of P with respect to O shall thus be as great as $O_F^{\leftarrow P}$ is big and as low as $P_F^{\leftarrow O}$ is small. The sets of attacks $O_F^{\leftarrow P}$ and $P_F^{\leftarrow O}$ may be arbitrarily large, so in order to construct a bounded acceptability scale, we transform their sizes $|O_F^{\leftarrow P}|$ and $|P_F^{\leftarrow O}|$ into values $x = f(|O_F^{\leftarrow P}|)$ and $y = f(|P_F^{\leftarrow O}|)$ using a monotonic increasing mapping $f : \mathbb{N} \to [0, 1[$ such that $f(0) = 0$ and $\lim_{n\to\infty} f(n) = 1$. The degree of acceptability may then be expressed as a function h of the variables x and y. Several choices are possible for such a function h, but the function $h(x, y) = \frac{1}{2}(1 + x - y)$ is remarkably the only one amongst those of the general form $h(x, y) = ax + by + c$ which fulfils simultaneously $h(x, y) = 1 - h(y, x)$ and $h(0, 1) = 0$. We thus adopt the following simple analytical expression

Definition 4 (degree of acceptability of P with respect to O)

$$\phi(P, O) = \frac{1}{2}\,[\,1 + f(\,|O_F^{\leftarrow P}|\,) - f(\,|P_F^{\leftarrow O}|\,)\,]$$

Concretely, for illustration purposes (and for the first part of proposition 5), we will use the mapping f defined $\forall n \in \mathbb{N}$ as

$$f(n) = 1 - \frac{1}{n+1} = \frac{n}{n+1}$$

For abstract the argumentation framework F shown in Fig. 1 and considering the opinions $P = \{a, c, f\}$ and $O = \{b, d, e\}$, the sets of attacks from P to O and from

O to P are respectively $O_F^{\leftarrow P} = \{(a,b),(c,d),(f,e)\}$, $P_F^{\leftarrow O} = \{(b,a),(e,c)\}$ and the degree of acceptability of P with respect to O is $\phi(P,O) = \frac{1}{2}[1+f(3)-f(2)] = \frac{1}{2}[1+\frac{3}{4}-\frac{2}{3}] = \frac{13}{24}$.

The notion of degree of acceptability can be used to define the notion of reward for the players in the given game of argumentation strategy. To properly defend the argument x, the proponent should naturally avoid self-contradiction. In other words, P should if possible be chosen so as to be conflict-free. Furthermore, to really play his role in the game, the opponent should contradict the proponent, therefore he should be maximally penalised whenever his opinion O fails to attack P. Finally, each player should seek to maximise the degree of acceptability of his opinion with respect to the one of his adversary. Rewards can be assigned to the players of the game of argumentation strategy in such a way as to give them a (material) incentive to follow these three fundamental principles of rationality.

Definition 5 (players' reward). *If P is not conflict-free, then the opponent should pay the proponent the sum $r_F(P,O) = 0$. If P is conflict-free and O does not attack P, then the opponent should pay the proponent the sum $r_F(P,O) = 1$. Otherwise, the opponent should pay the proponent the sum $r_F(P,O) = \phi(P,O)$.*

The following properties, which will only be used later (see Section 5), can be proved straightforwardly.

Proposition 1
1) $0 \leq r_F(P,O) \leq 1$
2.a) $r_F(P,O) = 0$ if and only if P is not conflict-free
2.b) $r_F(P,O) = 1$ if and only if P is conflict-free and O does not attack P
3) if P is admissible (or stable), then $r_F(P,O) \geq \frac{1}{2}[1 + f(|O|) - f(k\,|O|)]$, where k is the maximal out-degree (number of outgoing attacks) of the arguments contained in O
4) if there exist k attacks from O against P in F, then $r_F(P,O) < 1 - \frac{1}{2}\,f(k)$

According to definition 5, the proponent's reward is always equal to the opponent's loss. Games of argumentation strategy thus fall into the category of *zero-sum* games[1]. Note also that if the opponent fails to attack the proponent, then he is penalised with a maximal loss of 1. To reduce his loss, the opponent must then seek to minimise the number $|O_F^{\leftarrow P}|$ of attacks against his opinion O and maximise the number $|P_F^{\leftarrow O}|$ of attacks against the proponent's opinion P.

Finally, we impose that the players choose their strategy without prior knowledge of the strategy their adversary intends to play with. Games of argumentation strategy therefore also fall within the category of games with *imperfect information*. Since the outcome of a single round of an (F,x) game is random, in the nest section we will be exclusively interested in the game's outcome in the long run / after a large number of rounds – as is customary in Game Theory (Dresher 1981) when considering two-person zero-sum games with imperfect information.

[1] Such games have been extensively studied in the literature on Game Theory and used for analysing conflict situations in non-cooperative domains.

4 Strength of Arguments

In this section we define the proponent's long run expected payoff (the game's value) as a value of strength for the argument he embraces. Below, we explain how this value is mathematically defined and actually computed.

Intuitively, the proponent wants his reward $r_F(P, O)$ to be as large as possible, but he controls only the choice of P. The opponent wants to make his loss $r_F(P, O)$ as small as possible, but he only controls the choice of his strategy O. What are the guiding principles which should determine the player's choices and what is the expected outcome of such a game ?

As indicated at the end of section 3, each game of strategy needs to be repeated a large number of times. If a player were always to choose the same strategy, then his adversary could adapt his own strategy to it and get a better payoff. Therefore, it is important for players engaged in a repeated game with imperfect information to randomise their strategies over time. We therefore consider that each time the game is played, the proponent and opponent choose their strategies according to some probability distributions $X = (x_i)$ and $Y = (y_j)$. Thus, the probability of the proponent choosing his ith strategy P_i corresponds to x_i and the probability of the opponent choosing his jth strategy O_j corresponds to y_j. The probability distributions X and Y are called *mixed strategies*. If we denote by m and n the number of strategies available to the proponent and opponent respectively, then, to be valid distributions, X and Y must obviously be positive ($x_i, y_j \geq 0$) and sum up to 1 ($\sum_{i=1}^{m} x_i = \sum_{j=1}^{n} y_j = 1$).

By denoting the payoff matrix $R = ((r_{i,j}))_{m \times n}$ where $r_{i,j} = r_F(P_i, O_j)$ and by X^T the transpose of the m-dimensional vector $X = (x_i)$, the proponent's expected payoff is given by

$$E = X^T R Y = \sum_{j=1}^{n} \sum_{i=1}^{m} r_{i,j}\, x_i\, y_j$$

The proponent can therefore expect to get at least $\min_Y X^T R Y$, where the minimum is taken over all mixed strategies available to the opponent. Since the proponent has the choice of X, he will select X so that this minimum is as large as possible. Hence the proponent can pick a mixed strategy, denoted X^*, which will guarantee him an expectation of at least $\max_X \min_Y X^T R Y$ irrespective of what the opponent does. Similarly, the opponent can make the proponent's expected payoff at most equal to $\min_Y \max_X X^T R Y$ by playing with some strategy Y^*. The *minimax theorem* (von Neumann 1928) states that these two quantities always have a common value v

$$\max_X \min_Y X^T R Y = \min_Y \max_X X^T R Y = v$$

which is called the *value of the game*. This value is both the expected payoff that is guaranteed to the proponent and the maximal expected loss of the opponent. The strength measure we are after can be consequently defined as follows.

Definition 6 (argument strength). *The strength $s_F(x)$ of the argument x in the framework F is the value of the (F, x) game of argumentation strategy.*

Textbooks on Operations Research (Hillier and Lieberman 1995) explain how to compute v – when the game's value can be shown to be *a priori* positive – by solving a linear program with the simplex algorithm (Dantzig *et al.* 1955). It can be shown that v corresponds to the solution of the problem that consists in maximising the variable x_{m+1}, subject to the following $(n + m + 2)$ linear inequality constraints

$$\forall j \in \{1, \ldots, n\} : \sum_{i=1}^{m} r_{i,j}\, x_i - x_{m+1} \geq 0$$

$$\sum_{i=1}^{m} x_i = 1$$

$$x_1, \ldots, x_m, x_{m+1} \geq 0$$

Table 1. Strength (given in alphabetical order of the arguments) and ordering of arguments obtained in several abstract argumentation frameworks $F = (Arg, att)$

Ref.	Arguments Arg	Attacks att	Strength values	Ordering
F_1	{a}	{}	1	a
F_2	{a, b}	{(a,b)}	1, 0.25	a > b
F_3	{a, b}	{(a,b), (b,a)}	0.5, 0.5	a = b
F_4	{a,. . . , d}	{(a,b), (c,b), (d,c)}	1, 0.25, 0.25, 1	a = d > b = c
F_5	{a,. . . , d}	{(a,b), (c,b), (d,b), (b,d)}	1, 0.167, 1, 0.625	a = c > d > b
F_6	{a,. . . , e}	{(a,b), (c,b), (d,a), (e,c)}	0.25, 0.5, 0.25, 1, 1	d = e > b > a = c
F_7	{a,. . . , f}	{(a,b), (b,c), (c,d), (d,e), (e,f)}	1, 0.25, 0.5, 0.386, 0.5, 0.425	a = c = e > f > d > b
F_8	{a,. . . , f}	{(a,b), (b,a), (b,c), (c,d), (e,c), (f,e)}	0.5, 0.5, 0.417, 0.5, 0.25, 1	f > a = b = d > c > e

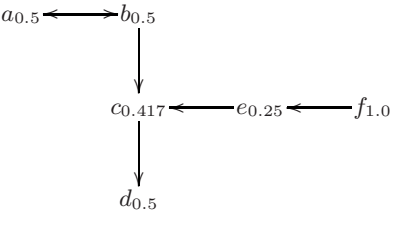


Fig. 2. Strength of arguments in F_8 in Table 1

Several examples of argument strength in elementary argumentation frameworks are provided in Table 1. For each one of these frameworks $F = (Arg, att)$ and each single argument x in them, we have constructed the (F, x) game payoff matrix $R = ((r_{i,j}))$ where $r_F(P_i, O_j)$ and computed the game's value using the simplex algorithm, as described above. The given ordering on the right-hand column provides a ranking over arguments, for each given framework. Fig. 2 shows the results obtained with framework F_8 (this is the framework already illustrated in Fig. 1). The proponent's optimal strategy [2] is to play $\{a, f\}$ for argument a,

[2] An optimal strategy is a (mixed) strategy with maximal expected payoff.

$\{b\}$ for b, $\{a, c, f\}$ with probability $\frac{2}{3}$ and $\{a, c\}$ with probability $\frac{1}{3}$ for c, $\{b, d\}$ for d, $\{b, e\}$ for e and $\{f\}$ for f. The ranking obtained is in decreasing order of strength $f > a = b = d > c > e$. This ranking mismatches with the ranking $f > d > a = b > e > c$ obtained using the measures proposed in (Besnard and Hunter 2001) and (Cayrol and Lagasquie-Schiex 2005). We regard the latter ranking as unintuitive, as it ranks argument e higher than argument c despite the facts that c is contained in the stable set of arguments $\{a, c, f\}$ and e is not even contained in an admissible set of arguments.

5 Core Properties of Argument Strength

In this section we conduct a thorough mathematical analysis of the properties of this game-theoretic measure of argument strength. Our first result shows that the argument strength scale is bounded.

Proposition 2 (bounds of argument strength). *The strength $s_F(x)$ of an argument x is such that $0 \leq s_F(x) \leq 1$.*

Proof. According to proposition 1, item 1), $\forall (i,j), r_{i,j} \in [0,1]$. For every mixed strategies X and Y, we also have $X^T R Y \in [0,1]$, which implies $0 \leq \min_Y X^T R Y$ and $\max_X X^T R Y \leq 1$. Therefore, $0 \leq \max_X \min_Y X^T R Y$ and $\min_Y \max_X X^T R Y \leq 1$. By the minimax theorem, $0 \leq v \leq 1$, and thus $v = s_F(x) \in [0,1]$.

The next two propositions show that the bounds found are both tight.

Proposition 3 (self-contradiction must be avoided). *The strength $s_F(x)$ of an argument x is 0 if and only if x attacks itself.*

Proof. $\Rightarrow$: $s_F(x) = v = \min_Y \max_X X^T R Y = 0$ implies the existence of Y^* such that $\forall X$, $X^T R Y^* \leq 0$. This holds notably for any $X = e_i$ (the vector whose components are all equal to 0 except the ith one which is equal to 1), hence, $\forall i$, $\sum_j r_{i,j} y_j^* \leq 0$. Since $r_{i,j} y_j^* \geq 0$, it is clear that $\forall (i,j)$, $r_{i,j} y_j^* = 0$. Y^* is a probability distribution, so there exists k such that $y_{j_k}^* > 0$. It is then necessary that $\forall i$, $r_{i,j_k} = 0$. According to proposition 1, item 2.a), $\forall i$, P_i attacks itself. In particular, $P_i = \{x\}$ attacks itself, *i.e.* argument x attacks itself.

$\Leftarrow$: If x attacks itself, then all proponent strategies in the (F, x) game are non-conflict-free sets of arguments. By proposition 1, item 2.a), $R = ((0))$ and $v = 0$.

Proposition 4 (unattacked arguments are the strongest). *The strength $s_F(x)$ of argument x is 1 if and only if there is no argument attacking x in F.*

Proof. $\Rightarrow$: If $s_F(x) = v = 1$, then we have $\max_X \min_Y X^T R Y = 1$. Y ranges over the set of all real-valued probability distributions which is larger than the set S of all zero-one valued probability distributions. Thus, $\forall X$, $\min_{Y \in S} X^T R Y \geq \min_Y X^T R Y$. Therefore, $\max_X \min_{Y \in S} X^T R Y \geq \max_X \min_Y X^T R Y = 1$. This can be rewritten as $\max_X \min_j \sum_i r_{i,j} x_i \geq 1$. $\exists X^*$ s.t. $\min_j \sum_i r_{i,j} x_i^* \geq 1$, *i.e.* $\forall j$, $\sum_i r_{i,j} x_i^* \geq 1$. Since $\forall (i,j)$, $r_{i,j} \leq 1$ and X^* is a probability distribution, $\forall j$, $\sum_i r_{i,j} x_i^* \leq 1$, so that in fact $\forall j$, $\sum_i r_{i,j} x_i^* = 1$. This may only hold if $\forall (i,j)$, $r_{i,j} < 1 \Rightarrow x_i^* = 0$. X^* is a probability distribution, so there exists k such that $x_k^* > 0$. By contraposition of the previous

implications, $\forall j$, $\neg(r_{k,j} < 1)$, *i.e.* $r_{k,j} \geq 1$. By proposition 1, item 1), $\forall j$, $r_{k,j} = 1$. By proposition 1, item 2.b), $\forall j$, P_k is conflict-free and O_j does not attack P_k. $x \in P_k$ so there is no opponent strategy or argument that attacks x.

$\Leftarrow$: By selecting strategy $\{x\}$ with probability 1, the proponent has a guaranteed payoff of 1 irrespective of what the opponent does. Therefore, $v \geq 1$. In fact, v is bounded up by 1 (by proposition 2) and $s_F(x) = 1$.

We can also show that admissible and stable arguments occupy the band of medium to high strength values [3] but that attacks against arguments reduce their strength below 1. We assume in part a) of the next proposition that f is defined, $\forall n \in \mathbb{N}$, by $f(n) = \frac{n}{n+1}$ as suggested in Section 3.

Proposition 5 (acceptable arguments have medium to high strength). *a) If there exists an admissible (or stable) set of arguments containing x, then $s_F(x) \geq \frac{1}{2}[1 + \frac{1}{2} - \frac{k}{k+1}]$ where k is the maximal out-degree of arguments in F. b) If there exist n attacks against x, then $s_F(x) < 1 - \frac{1}{2}f(n)$.*

Proof. a) If P is admissible, then by proposition 1, item 3), $\forall O$, $r_F(P, O) \geq \frac{1}{2}[1 + f(|O|) - f(k\,|O|)] \geq \frac{1}{2}[1 + f(1) - f(k)]$ (when $f(n) = \frac{n}{n+1}$). By playing P with probability 1 the proponent of x can secure a payoff of at least $\frac{1}{2}[1 + \frac{1}{2} - \frac{k}{k+1}]$. If P is stable, then P is also admissible and the same inequality holds. b) If there exist n attacks against x, then there exists an opponent strategy O with n attacks against x. For this strategy, and whatever the proponent strategy P, there must also exist at least n attacks from O against P and $r_F(P, O) < 1 - \frac{1}{2}f(n)$ by proposition 1, item 4). By playing O with a probability of 1, the opponent can secure a strict maximum loss of $1 - \frac{1}{2}f(n)$.

Note that when the maximal out-degree in F is $k = 1$, the strength of acceptable arguments is greater than $\frac{1}{2}$, hence the use of the term "medium".

We now study how the strength of arguments varies as argumentation frameworks are expanded. This should allows us to understand quantitatively the impact of adding new arguments and attacks to a dispute. Then, suppose first that we add an attack (a, b) to the framework $F = (Arg, att)$, where $(a, b) \notin att$ and $a, b \in Arg$. For convenience, in this case we adopt

Notation 2. $F_{+(a,b)} = (\, Arg,\, att \cup \{(a, b)\}\,)$

As intuitively expected, adding an attack against an argument reduces its strength:

Proposition 6 (criticism reduces argument strength). $s_{F_{+(a,b)}}(b) \leq s_F(b)$.

Proof. The sets of strategies available to the proponent and opponent are the same in the (F, b) and $(F_{+(a,b)}, b)$ games. Let P and O be proponent and opponent strategies. Note that $P_F^{\leftarrow O} \subseteq P_{F_{+(a,b)}}^{\leftarrow O}$ and either $O_F^{\leftarrow P} = O_{F_{+(a,b)}}^{\leftarrow P}$ (if $a \notin P$) or P attacks itself in $F_{+(a,b)}$ (if $a \in P$). By monotonicity of f, $\phi_{F_{+(a,b)}}(P, O) \leq \phi_F(P, O)$. In any case ($a \in P$ or $a \notin P$), $r_{F_{+(a,b)}}(P, O) \leq r_F(P, O)$. It follows that $s_{F_{+(a,b)}}(b) \leq s_F(b)$.

[3] This property can be generalised to any notion of acceptability "stronger" than admissibility, such as *e.g.* the preferred, complete, grounded and ideal semantics (Dung 1995, Bondarenko *et al.* 1997, Dung *et al.* 2006, 2007).

Adding an attack from argument a against b gives an advantage to the proponent of a as long as b is not useful in the defence of a. Otherwise, this new attack constitutes a handicap for the proponent. To distinguish between these two possible cases, we say that

Definition 7 (superfluous argument). *Argument b is superfluous with respect to a if forbidding the proponent of a to play with strategies containing b does not decrease the proponent's payoff in the (F, a) game (the game's value).*

Proposition 7 (cautious extra-aggressiveness increases strength). *By adding an attack (a, b) one increases a's strength ($s_{F_{+(a,b)}}(a) \geq s_F(a)$) if b is superfluous with respect to a and diminishes it ($s_{F_{+(a,b)}}(a) \leq s_F(a)$) otherwise.*

Proof. If b is superfluous with respect to a then there exists an optimal mixed strategy X^* for the (F, a) game such that $\forall i, x_i^* > 0 \Rightarrow b \notin P_i$. Let then P be an active strategy, *i.e.* $P = P_i$ and $x_i^* > 0$. Then, $\forall O$, we have $O_F^{\leftarrow P} \subseteq O_{F_{+(a,b)}}^{\leftarrow P}$, $P_F^{\leftarrow O} = P_{F_{+(a,b)}}^{\leftarrow O}$ (if it is not the case that $a \in O$ and $b \in P$) or P attacks itself in $F_{+(a,b)}$ (if $a \in O$ and $b \in P$). The last case does not occur ($b \notin P$) since b is assumed to be superfluous with respect to a. By monotonicity of f, $\phi_F(P, O) \leq \phi_{F_{+(a,b)}}(P, O)$. Since $b \notin P$, P is conflict-free in F iff P is conflict-free in $F_{+(a,b)}$ and O attacks P in F iff O attacks P in $F_{+(a,b)}$. Therefore, for every active strategy P under X^* we have $r_F(P, O) \leq r_{F_{+(a,b)}}(P, O)$. By playing with X^* in the $(F_{+(a,b)}, a)$ game, the proponent can secure a payoff of at least $s_F(a)$. Hence, $s_{F_{+(a,b)}}(a) \geq s_F(a)$. If b is not superfluous with respect to a, then the proponent of a is forced (otherwise his payoff is null) to play strategies containing a but not b, and thus his payoff is reduced.

Moreover, the strength of an argument x may be partially restored by adding an attack (a, b) against one of its attackers b.

Proposition 8 (indirect counter-attack brings support). *If b attacks x, adding an attack (a, b) to F increases x's strength ($s_{F_{+(a,b)}}(x) \geq s_F(x)$).*

Proof. The sets of strategies of the players are the same in the (F, x) and $(F_{+(a,b)}, x)$ games. We have $O_F^{\leftarrow P} \subseteq O_{F_{+(a,b)}}^{\leftarrow P}$ (if $a \in P$ and $b \in O$) or $O_F^{\leftarrow P} = O_{F_{+(a,b)}}^{\leftarrow P}$ otherwise. We also have $P_F^{\leftarrow O} \subseteq P_{F_{+(a,b)}}^{\leftarrow O}$ (if $b \in P$ and $a \in O$) and $P_F^{\leftarrow O} = P_{F_{+(a,b)}}^{\leftarrow O}$ otherwise. Note that if $b \in P$ then P attacks itself in both F and $F_{+(a,b)}$. So, $r_F(P, O) \leq r_{F_{+(a,b)}}(P, O)$ and $s_F(x) \leq s_{F_{+(a,b)}}(x)$.

So far we have considered adding attacks between existing arguments in a given dispute (argumentation framework). We finally consider adding new arguments, and show that, as intuitively expected, the status of arguments in this dispute is left unchanged if the newly added arguments are "disconnected" from the original ones.

Proposition 9 (insensitivity to irrelevant information). *If $F' = (Arg', att')$ is such that $Arg \cap Arg' = \emptyset$, then $s_{F+F'}(x) = s_F(x)$ where $F + F' = (Arg \cup Arg', att \cup att')$.*

Proof. Let us consider the $(F + F', x)$ game where $x \in Arg$. Since no argument in Arg' attacks x (the two frameworks are disconnected), the proponent of x is at least

as well off in this new game as in the (F, x) game if he restricts himself to his old set of strategies build only from Arg. Therefore, $s_{F+F'}(x) \geq s_F(x)$. The same proposition also holds for the opponent of x, which means that $-s_{F+F'} \geq -s_F(x)$ or equivalently $s_{F+F'}(x) \leq s_F(x)$. In conclusion, $s_{F+F'}(x) = s_F(x)$.

6 Conclusion

Arguments, opinions and conflicts between opinions in disputes can conveniently be modelled using abstract argumentation frameworks (Dung 1995). In order to assess the strength of an argument in a dispute, we defined a repeated game of argumentation strategy whereby two players, *viz.* the proponent and opponent of the argument, simultaneously exchange sets of arguments representing respectively opinions for and against it. We defined a degree of acceptability and reward function for a single round based on the intuition that it is better to have more attacks on and fewer attacks from adversarial opinions within the framework representing the dispute considered. Then, players choose their opinions randomly in each round with a certain probability so as to maximise their expected reward in the long run. The strength of the argument of interest is finally determined by the value of the proposed game (von Neumann 1928) as defined in Game Theory for two-person zero-sum games with imperfect information and computed using the simplex algorithm (Dantzig *et al.* 1955).

We have shown that such a measure of strength is bounded (between 0 and 1), that these bounds are attained for arguments that respectively attacks themselves and that are not attacked, and also that arguments contained into admissible or stable extensions (Dung 1995, Bondarenko *et al.* 1997, Dung 2006, 2007) always have medium to high strength values, but that attacks against such acceptable arguments reduce their strength value below 1. This result also holds for preferred, complete, grounded or ideal extensions, but we have omitted its proof here for lack of space. We have examined the sensitivity of strength values with respect to changes operated on the underlying argumentation framework. Notably, we have seen that adding a new attack against an argument reduces its strength, that adding a new attack against another argument either increases or reduces its strength, depending on the usefulness of the target in the defence of that argument, and that the addition of indirect counter-attacks could restore the strength of an argument. Finally, we have proved that the addition of irrelevant groups of arguments to a dispute does not influence the status of its original arguments.

Several notions of argument strength have already been proposed in the literature on argumentation. One shall distinguish between the so-called "intrinsic" and "interaction-based" measures. The term *intrinsic* is used to refer to approaches whereby the strength of an argument is independent of its interaction with other arguments (Pollock 1992, Krause *et al.* 1995, Ambler 1996, Parsons 1997, Prakken and Sartor 1997, Amgoud and Cayrol 1998, Kohlas *et al.* 2000, Pollock 2001). On the other hand, the term *interaction-based* refers to measures whereby the strength of an argument depends on the arguments attacking it (attackers), the attackers of its attackers (the defenders), *etc.* Amongst

interaction-based measures of argument strength, one may again distinguish between qualitative (Dung 1995, Jakobovits and Vermeir 1999) and quantitative (Besnard and Hunter 2001, Cayrol and Lagasquie-Schiex 2005) measures.

Our interaction-based and quantitative measure is thus closest to the approaches by (Besnard and Hunter 2001) and (Cayrol and Lagasquie-Schiex 2005). However, we have observed (see last paragraph of Section 4), that these measures may produce a different ranking of arguments than ours, and have argued that in general they do not convey the intuition according to which the dialectical properties of admissibility and stability should confer more strength to an argument in a dispute. These measures thus somewhat transgress the principles of dialectic originally proposed by (Dung 1995).

There is other work in the field of argumentation which directly relates to Game Theory. This typically involves "extensive" games, namely multi-stage games that can be represented as trees. Argumentation games in extensive form have been proved to be useful to test the acceptability of arguments under various semantics (Vreeswijk and Prakken 2000, Dunne and Bench-Capon 2003) and to determine optimal strategies (Riveret *et al.* 2008) in dialogues (Prakken 2005). To this date and to the best of our knowledge, the only other work in the domain of argumentation that has sought to exploit strategic games is (Rahwan and Larson 2008), but to model argumentation between self-interest agents.

Acknowledgements. This work was funded by the Sixth Framework IST programme of the EC, under the 035200 ArguGRID project. We wish to express our gratitude to the anonymous referees for their helpful comments and suggestions.

References

Ambler, S.J.: A Categorical Approach to the Semantics of Argumentation. Mathematical Structures in Computer Science 6(2), 167–188 (1996)

Amgoud, L., Cayrol, C.: On the acceptability of arguments in preference-based argumentation. In: 14th Conference on Uncertainty in Artificial Intelligence (UAI 1998), pp. 1–7 (1998)

Amgoud, L., Prade, H.: Using Arguments for Making Decisions: A Possibilistic Logic Approach. In: 20th Conference of Uncertainty in Artificial Intelligence (UAI 2004), pp. 10–17 (2004)

Amgoud, L., Dupin de Saint-Cyr, F.: Measures for Persuasion Dialogs: A Preliminary Investigation. In: 2nd International Conference on Computational Models of Argument (COMMA 2008), pp. 13–24 (2008)

Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. Artificial Intelligence 128, 203–235 (2001)

Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artificial Intelligence 93(1-2), 63–101 (1997)

Borel, E.: La théorie du jeu et les équations intégrales à noyau symétrique gauche. Comptes Rendus de l'Académie des Sciences (1921)

Budzyńska, K., Kacprzak, M., Rembelski, P.: Modelling Persuasiveness: Change of Uncertainty Through Agents' Interactions. In: 2nd International Conference on Computational Models of Argument (COMMA 2008), pp. 85–96 (2008)

Cayrol, C., Lagasquie-Schiex, M.-C.: Graduality in Argumentation. Journal of Artificial Intelligence Research 23, 245–297 (2005)

Dantzig, G.B., Orden, A., Wolfe, P.: The generalised simplex method for minimizing a linear form under linear inequality constraints. Pacific Journal of Mathematics 5(2), 183–195 (1955)

Dresher, M.: The Mathematics of Games of Strategy. Dover Publications (1981)

Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. Artificial Intelligence 77(2), 321–357 (1995)

Dung, P.M., Kowalski, R., Toni, F.: Dialectic proof procedures for assumption-based, admissible argumentation. Artificial Intelligence 170(2), 114–159 (2006)

Dung, P.M., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artificial Intelligence 171, 642–674 (2007)

Dunne, P.E., Bench-Capon, T.J.M.: Two party immediate response disputes: Properties and efficiency. Artificial Intelligence 149, 221–250 (2003)

Hillier, F.S., Lieberman, G.J.: Introduction to Operations Research, 6th edn. McGraw-Hill, New York (1995)

Jakobovits, H., Vermeir, D.: Robust semantics for argumentation frameworks. Journal of logic and computation 9(2), 215–261 (1999)

Kohlas, J., Haenni, R., Berzati, D.: Probabilistic argumentation systems and abduction. In: 8th International Workshops on Non-Monotonic Reasoning, pp. 391–398 (2000)

Krause, P., Ambler, S., Elvang-Gøransson, M., Fox, J.: A logic of argumentation for reasoning under uncertainty. Computational Intelligence 11, 113–131 (1995)

Parsons, S.: Normative argumentation and qualitative probability. In: 1st International Joint Conference on Qualitative and Quantitative Practical Reasoning, pp. 466–480 (1997)

Pollock, J.L.: How to reason defeasibly. Artificial Intelligence 57, 1–42 (1992)

Pollock, J.L.: Defeasible reasoning with variable degrees of justification. Artificial Intelligence 133, 233–282 (2001)

Poole, D.: Probabilistic Horn Abduction and Bayesian Networks. Artificial Intelligence 64, 81–129 (1993)

Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. Journal of Applied Non Classical Logics 7, 25–75 (1997)

Prakken, H.: Coherence and Flexibility in Dialogue Games for Argumentation. Journal of Logic and Computation 15(6), 1009–1040 (2005)

Rahwan, I., Larson, K.: Mechanism Design for Abstract Argumentation. In: 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), pp. 1031–1038 (2008)

Riveret, R., Prakken, H., Rotolo, A., Sartor, G.: Heuristics in Argumentation: A Game-Theoretical Investigation. In: 2nd International Conference on Computational Models of Argument (COMMA 2008), pp. 324–335 (2008)

Vreeswijk, G., Prakken, H.: Credulous and sceptical argument games for preferred semantics. In: Brewka, G., Moniz Pereira, L., Ojeda-Aciego, M., de Guzmán, I.P. (eds.) JELIA 2000. LNCS (LNAI), vol. 1919, pp. 239–253. Springer, Heidelberg (2000)

von Neumann, J.: Zur Theorie des Gesellschaftsspiele. Mathematische Annalen 100, 295–320 (1928)

von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, Princeton (1944)

A Tableau for RoBCTL*

John C. McCabe-Dansted*

University of Western Australia
john@csse.uwa.edu.au

Abstract. It can be desirable to specify polices that require a system to achieve some outcome even if a certain number of failures occur. The temporal logic of robustness RoCTL* extends CTL* with operators from Deontic logic, and a novel operator referred to as "Robustly" [7]. The only known decision procedure for RoCTL* involves a reduction to QCTL*. This paper presents a tableau for the related bundled tree logic RoBCTL*; this is the first decision procedure for RoBCTL*, and although non-elementary in degenerate cases has much better performance than the QCTL* based decision procedure for RoCTL*. The degenerate cases where this tableau performs poorly provide clues as to where we might look to prove that RoBCTL* is non-elementary.

Keywords: Logic, Tableau, Robustness, Bundled, Diagnosis, QCTL*.

1 Introduction

The RoCTL* logic [11,7] is an extension of CTL* introduced to represent issues relating to robustness and reliability in systems. It does this by adding an Obligatory operator and a Robustly operator. The Obligatory operator specifies how the systems should behave by quantifying over paths in which no failures occur. The Robustly operator specifies that something must be true on the current path and similar paths that "deviate" from the current path, having at most one more failure occurring. This notation allows phrases such as "even with n additional failures" to be built up by chaining n simple unary Robustly operators together. We have previously given examples of robust systems that can be concisely represented in RoCTL* [7].

The RoCTL* Obligatory operator is similar to the Obligatory operator in Standard Deontic Logic (SDL), although in RoCTL* the operator quantifies over paths rather than worlds. SDL has many paradoxes. Some of these, such as the "Gentle Murderer" paradox spring from the inadequacy of SDL for dealing with obligations caused by acting contrary to duty such as "If you murder, you must murder gently". Contrary-to-Duty (CtD) obligations are important for modeling

* The author would like to thank Mark Reynolds and Tim French for their valuable feedback on the construction of the tableau presented in this paper.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 298–310, 2008.

a robust system, as it is often important to state that the system should achieve some goal and also that if it fails it should in some way recover from the failure. RoCTL* can represent CtD obligations by specifying that the agent must ensure that the CtD obligation is met even if a failure occurs. For further discussion of CtD obligations and motivations for RoCTL*, see [7].

Unfortunately the only known decision procedure for RoCTL* [7] involves a reduction to QCTL* which is non-elementary to decide. Attempts to find an automaton based decision procedure for RoCTL* were hampered by the importance that replacing path quantifiers with atoms plays in existing automaton based decision procedures [4] for CTL*. In RoCTL* the robustly operator is a path quantifier, but unlike the "All paths" operator, a formula starting with "robustly" is not a state formula and thus cannot be replaced with an atom. For this reason research has focused on finding resolution and tableau based decision procedures. Unfortunately there are currently no known resolution or tableau decision procedures for CTL*.

A CTL like restriction RoCTL was proposed with a resolution based decision procedure [3]. In this paper we present a more expressive logic RoBCTL*, a bundled tree variant of RoCTL*, and extend Reynolds' [12] tableau for BCTL* to decide RoBCTL*. This decision procedure is the first for RoBCTL* as the reduction to QCTL* used for RoCTL* does not work for RoBCTL* as we know of no decision procedure for bundled variants of QCTL*.

Two features of the RoBCTL* tableau not present in the BCTL* tableau are the ability to deal with path quantifiers that do not result in state formulae and the ability to deal with eventualities that change over time. With QCTL* we implemented the robustly operator by marking the current path. Our attempts to extend the BCTL* tableau to allow marked paths, or bundles of deviating paths resulted in non-finite tableaux. We have instead used a successor function that adds enough formulae to the closure that we can handle deviations without having to distinguish paths in the Tableau. For example, the successor of "Next A" is "A". This can make the closure set large, although the size of the closure set will be elementary if the alternations between Robustly and Until are bounded.

Another feature not present in the BCTL* tableau is the ability to deal with eventualities that change over time. In BCTL* the only eventuality is of the form "A Until B", which remains unchanged until it is resolved by B occurring. In RoBCTL* we have "Eventually a path will deviate and along that path A will be hold", at the next step this becomes "... the successor of A will hold".

There is currently no axiomatisation of RoBCTL* or RoCTL* that is known to be complete. An interesting question is whether the robustly operator can be expressed in CTL* and BCTL*. It is not known if this is the case, as will be discussed in this paper.

A number of other extensions of temporal logics have been proposed to deal with Deontic or Robustness issues [2,9,8,1,13]. As noted previously [7], each of these logics are substantially different from RoCTL*. Additionally RoBCTL* logic is the first such logic to use the bundled tree semantics.

2 RoBCTL* Logic

2.1 RoBCTL* Syntax

As with the RoCTL* Logic, the RoBCTL* logic has a set $\mathcal{V}$ of atomic propositions that we call variables, including a special atom Viol. Where p varies over $\mathcal{V}$, we define formulæ according to the following abstract syntax

$$\phi := \top \mid p \mid \neg\phi \mid (\phi \wedge \phi) \mid (\phi U \phi) \mid N\phi \mid A\phi \mid O\phi \mid \blacktriangle\phi \ .$$

For consistency with [7], we do not consider a formula that explicitly contains Viol to be a RoBCTL* formula, although this tableau makes use of such formulae internally and can work with input formulae that contain Viol.

The $\top$, $\neg$, $\wedge$, N, U and A are the familiar "true", "not", "and", "next", "until" and "all paths" operators from CTL. The abbreviations $\bot$, $\vee$, F, G, W, E $\rightarrow$ and $\leftrightarrow$ are defined as in CTL* logic. As with Standard Deontic Logic (SDL) logic, we define $P \equiv \neg O \neg$. Finally, we define the abbreviation $\Delta \equiv \neg\blacktriangle\neg$.

The formula $\blacktriangle\phi$ is read as "robustly ϕ" and means roughly "ϕ is true and will remain true if a deviation occurs". The formula $O\phi$ is read as "obligatory ϕ" and means roughly "ϕ is true if no failures occur".

2.2 RoCTL-Structures

Definition 1. *We say that a binary relation R on S is serial (total) if for every a in S there exists b in S such that aRb.*

A transition frame is a pair $(A, \rightarrow)$, where A is a non-empty set of states and $\rightarrow$ is a serial relation on A.

Definition 2. *We let $\mathcal{V}$ be our set of variables. The set $\mathcal{V}$ contains a special variable Viol. A valuation g is a map from a set of states A to the power set of the variables. The statement $p \in g(w)$ means roughly "the variable p is true at state w".*

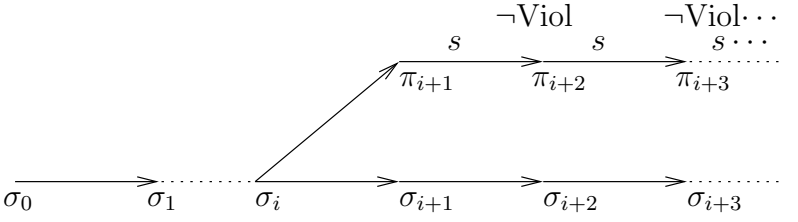
Definition 3. *We call an ω-sequence $\sigma = \langle w_0, w_1, \ldots\rangle$ of states a fullpath iff for all non-negative integers i we have $w_i \rightarrow w_{i+1}$. For all i in $\mathbb{N}$ we define $\sigma_{\geq i}$ to be the fullpath $\langle w_i, w_{i+1}, \ldots\rangle$, we define σ_i to be w_i and we define $\sigma_{\leq i}$ to be the sequence $\langle w_0, w_1, \ldots, w_i\rangle$. We say that a set of fullpaths B is fusion closed iff for all non-negative integers i, j and $\sigma, \pi \in B$ we have $\langle \sigma_0, \sigma_1, \ldots, \sigma_i, \pi_j, \pi_{j+1}, \ldots\rangle \in B$ if $\sigma_{i+1} = \pi_j$. We say that a set of fullpaths B is suffix closed iff for all integers i and $\sigma \in B$ we have $\sigma_{\geq i} \in B$. We say a set of fullpaths is a bundle if it is non-empty, suffix closed and fusion closed.*

Definition 4. *A BCTL-structure $M = (A, \rightarrow, g, B)$ is a 4-tuple containing a set of states A, a serial binary relation $\rightarrow$, a valuation g on the set of states A, and B is a bundle on $(A, \rightarrow)$.*

Definition 5. *We say that a fullpath σ is failure-free iff for all $i > 0$ we have $Viol \notin g(\sigma_i)$. We define $S\!F(w)$ to be the set of all fullpaths in B starting with w and $S(w)$ to be the set of all* failure-free *fullpaths in B starting with w. We call a BCTL structure a RoBCTL structure iff $S(w)$ is non-empty for every $w \in A$.*

Definition 6. *For two fullpaths σ and π we say that π is an i-deviation from σ iff $\sigma_{\leq i} = \pi_{\leq i}$ and $\pi_{\geq i+1} \in S(\pi_{i+1})$. We say that π is a deviation from σ if there exists a non-negative integer i such that π is an i-deviation from σ. We define a function δ from fullpaths to sets of fullpaths such that where σ and π are fullpaths, a fullpath $\pi \in B$ is a member of $\delta(\sigma)$ iff π is a deviation from σ.*

We see that $S(\sigma_0) \subseteq \delta(\sigma) \subseteq \mathcal{SF}(\sigma_0)$. Below is an example of an i-deviation π from a fullpath σ. The arrows labeled with s represents transitions that must be successful, and are followed by nodes labelled ¬Viol.



2.3 RoBCTL* Semantics

We define truth of a RoBCTL* formula ϕ on a fullpath $\sigma = \langle w_0, w_1, \ldots \rangle$ in a BCTL-structure M recursively as follows:

$$
\begin{aligned}
M, \sigma &\vDash N\phi \text{ iff } M, \sigma_{\geq 1} \vDash \phi \\
M, \sigma &\vDash \phi U \psi \text{ iff } \exists_{i \in \mathbb{N}} \text{ s.t. } M, \sigma_{\geq i} \vDash \psi \text{ and } \forall_{j \in \mathbb{N}} j < i \implies M, \sigma_{\geq j} \vDash \psi \\
M, \sigma &\vDash A\phi \text{ iff } \forall_{\pi \in \mathcal{SF}(\sigma_0)} M, \pi \vDash \phi \\
M, \sigma &\vDash O\phi \text{ iff } \forall_{\pi \in S(\sigma_0)} M, \pi \vDash \phi \\
M, \sigma &\vDash \blacktriangle\phi \text{ iff } \forall_{\pi \in \delta(\sigma)} M, \pi \vDash \phi \text{ and } M, \sigma \vDash \phi\,.
\end{aligned}
$$

The definitions for $\top$, p, $\neg$ and $\wedge$ are as we would expect from classical logic. We say that a formula ϕ is valid in RoBCTL* iff for all RoBCTL structures $M = (A, \rightarrow, g, B)$, for all fullpaths σ in B we have $M, \sigma \vDash \phi$.

2.4 An Interesting Question

It is clear that if we expose the Viol atom to the logic, the O operator becomes redundant, as we can write $O\phi$ as $A(GN\neg\text{Viol} \rightarrow \phi)$. We do not know if we can likewise remove the ▲ operator. We have shown that we can remove the ▲ operator if we introduce the ∃ and ∀ operators from QCTL* [7]. We can translate some simple formulae into (B)CTL*. For example, $\blacktriangle Gp$ becomes

$$pW(E(GN\neg\text{Viol} \wedge Gp)).$$

Since the proposal of this logic [11], one of our primary research goals has been attempting to show that we can express Ro(B)CTL* in (B)CTL*. We have attempted to find a recursive translation f that removes ▲ operators. Translating conjunctions is non-trivial, we cannot let $f(\Delta(\phi \wedge \psi)) = f(\Delta\phi) \wedge f(\Delta\psi)$ as

this would allow ϕ and ψ to hold on two different deviations. For simple formulae we can divide the formulae into stages, and enumerate the stages, e.g. if we are taking the conjunction of aUb and cUd we can consider three cases: b occurs before d, b occurs together with d and b occurs after d. It is then easy to take the conjunction of each stage. This becomes tricky with nested untils, e.g. $(aUb)Uc$, as stages can occur in cycles and it can become difficult to find a finite enumeration of all possibilities in a form that can be expressed in CTL*.

Another approach we have considered is to take a translation into QCTL*, and attempt to rearrange the formula to remove the $\exists$ and $\forall$ operators. We have also considered taking the translation for CTL* into automata, modifying the automata so as to express RoCTL*. However, we have not found any way of translating the automata back into CTL* Given that we have considered several translations into (B)CTL* and each had flaws we could not fix, we conjecture that no correct translation exists.

Conjecture 1. There exist RoBCTL* formulae that cannot be expressed in BCTL*. Likewise, there exist RoCTL* formulae that cannot be expressed in CTL*.

3 A Tableau for RoBCTL*

Here we define a tableau RoBCTL-TAB for deciding RoBCTL*. This tableau is derived from Reynolds' [12] tableau for BCTL*.

Definition 7. *We define logical operations on sets of formulae as follows:*

$$(S * T) = \{\varepsilon : \exists_{\phi \in S} \exists_{\psi \in T} \ s.t. \ \varepsilon = (\phi * \psi)\} \ \ where * \in \{U, \wedge\}$$
$$*S = \{\varepsilon : \exists_{\phi \in S} \ s.t. \ \varepsilon = *\phi\} \ \ where * \in \{\neg, \blacktriangle, A, N, O\}$$

We define the abbreviations $\Delta, E, P, \vee, \rightarrow, \leftrightarrow$ on sets similarly.

Definition 8. *We let Ξ be a formula translation function such that $\Xi(\phi) = \Xi(\psi)$ iff ϕ is equivalent to ψ under classical logic taking all subformulae with non-classical operators of highest precedence as atoms. Likewise we define Ξ on sets of formulae such that $\Xi(\Phi) = \{\Xi(\phi) : \phi \in \Phi\}$. We leave the choice of Ξ as an implementation detail. We could choose Ξ such that it simply normalises all formulae into clausal form. However to improve human understanding of the tableau, we may wish to choose Ξ such that it maximises the number of cases where $\Xi(\phi) = \phi$ and/or attempts to produce short formulae.*

Definition 9. *Below we define a function N^{-1} from formulae to sets of formulae. The intention is that N^{-1} is in some sense the inverse of prefixing a formula with N and thus that $A\left(\phi \leftrightarrow NN^{-1}\phi\right)$ is true. However, $(p \leftrightarrow N\top)$ is true if p is true whereas $(p \leftrightarrow N\bot)$ is true if p is false. Thus the "inverse" of N depends on the truth of state formulae at the current state, and so N^{-1} returns a set rather than just a single formula.*

Definition 10. *We define a function* N^{-1} *from formulae to sets of formulae:*

$$\begin{aligned} N^{-1}(\phi U \psi) &= \left(N^{-1}(\phi) \wedge \{(\phi U \psi)\}\right) \vee N^{-1}(\psi) \\ N^{-1}(\neg\phi) &= \neg N^{-1}(\phi) \\ N^{-1}(N\phi) &= \{\phi\} \\ N^{-1}(\phi \wedge \psi) &= N^{-1}(\phi) \wedge N^{-1}(\psi) \\ N^{-1}(p) &= \{\top, \bot\} \\ N^{-1}(A\phi) &= \{\top, \bot\} \\ N^{-1}(O\phi) &= \{\top, \bot\} \\ N^{-1}(\blacktriangle\phi) &= \{\bot\} \cup \left(\blacktriangle \Xi\left(N^{-1}(\phi)\right)\right) \\ N(\bot) &= \{\bot\} \end{aligned}$$

We extend N^{-1} to operate on sets of formulae as follows: given a set of formulae Φ, a formula ψ is a member of $N^{-1}(\Phi)$ iff there exists $\phi \in \Phi$ such that $\psi \in N^{-1}(\phi)$.

Definition 11. We define N^{-i} recursively as $N^{-i}(\Phi) = N^{-1}\left(N^{1-i}(\Phi)\right)$ and $N^0(\Phi) = \Phi$. Let $N^\star(\Phi)$ be the normalised closure of a set of formulae Φ under N^{-1}. That is, $\phi \in N^\star(\Phi)$ iff *there exists a non-negative integer* i *such that* $\phi \in \Xi\left(N^{-i}(\Phi)\right)$

Although $N^\star$ and N^{-1} are finite, they can become very large. See 5 for a detailed discussion of cardinality.

Definition 12. *For any pair of formulae* (ϕ, ψ)*, we say that* $\phi \leq \psi$ *iff* ϕ *is a subformula of* ψ*.*

Definition 13. *Let* $\gamma = NG\neg Viol$ *represent the statement "this path is failure-free" (this statement is not a RoBCTL* formula because it contains Viol). The closure* $\mathbf{cl}\phi$ *of the formula* ϕ *is defined as the smallest set that satisfies the four following requirements:*

1. $\mathbf{cl}\phi \supseteq \{\phi, \gamma\}$
2. For all $\psi \in \mathbf{cl}\phi$, if $\delta \leq \psi$ then $\delta \in \mathbf{cl}\phi$.
3. For all $\psi \in \mathbf{cl}\phi$, $\neg\psi \in \mathbf{cl}\phi$ or there exists δ such that $\psi = \neg\delta$ and $\delta \in \mathbf{cl}\phi$.
4. If $\blacktriangle\psi \in \mathbf{cl}\phi$ then $\mathbf{cl}\phi \supseteq \blacktriangle N^\star(\psi)$ and $\mathbf{cl}\phi \supseteq ANON^\star(\psi)$.

Recall that we have defined logical operations on sets of formulae (Definition 7). Thus $ANON^\star(\psi)$ represents a set of formulae, each prefixed with ANO.

The requirement (4) above is required to ensure that the successor formulae from Definitions 10 and 16 are included in the closure set.

Definition 14 (MPC). *We say that* $a \subseteq \mathbf{cl}\phi$ *is* MPC *iff for all* $\alpha, \beta \in a$

(M1) if $\beta = \neg\alpha$ then $\beta \in a$ iff $\alpha \notin a$,
(M2) if $\alpha \wedge \beta \in \mathbf{cl}\phi$ then $(\alpha \wedge \beta) \in a \leftrightarrow (\alpha \in a$ and $\beta \in a)$

A hue is roughly speaking a set of formulae that could hold along a single fullpath.

Definition 15 (Hue). *A set $a \subseteq \mathbf{cl}\phi$ is a hue for ϕ iff*

(H1) a is MPC;
(H2) if $\alpha U \beta \in a$ then $\alpha \in a$ or $\beta \in a$;
(H3) if $\neg(\alpha U \beta) \in a$ then $\beta \notin a$; and
(H4) if $A\alpha \in a$ or $\blacktriangle\alpha \in a$ then $\alpha \in a$.

Let H_ϕ be the set of hues of ϕ.

Definition 16. *For each hue a in H_ϕ, we define a formula translation function N_a^{-1}. Note that $N_a^{-1}(\phi) \in N^{-1}(\phi)$.*

$$
\begin{aligned}
N_a^{-1}(\phi U \psi) &= \left(N_a^{-1}(\phi) \wedge (\phi U \psi)\right) \vee N_a^{-1}(\psi) \\
N_a^{-1}(\neg\phi) &= \neg N_a^{-1}(\phi) \\
N_a^{-1}(N\phi) &= \phi \\
N_a^{-1}(\phi \wedge \psi) &= N_a^{-1}(\phi) \wedge N_a^{-1}(\psi) \\
N_a^{-1}(p) &= \begin{cases} \bot & \text{if } p \notin a \\ \top & \text{if } p \in a \end{cases} \\
N_a^{-1}(A\phi) &= \begin{cases} \bot & \text{if } A\phi \notin a \\ \top & \text{if } A\phi \in a \end{cases} \\
N_a^{-1}(O\phi) &= \begin{cases} \bot & \text{if } O\phi \notin a \\ \top & \text{if } O\phi \in a \end{cases} \\
N_a^{-1}(\blacktriangle\phi) &= \begin{cases} \bot & \text{if } ANO\Xi\left(N_a^{-1}(\phi)\right) \notin a \\ \blacktriangle\Xi\left(N_a^{-1}(\phi)\right) & \text{otherwise} \end{cases}
\end{aligned}
$$

Definition 17. *We define a function h on paths such that*

$$h(\pi) = \{\alpha : \alpha \in \mathbf{cl}\phi \text{ and } \pi \vDash \alpha\}$$

From the semantics of RoBCTL*, we see that for each $\pi \in B$, $h(\pi)$ is a hue.

Lemma 1. *If $h(\pi) = a$ and $\theta_0 = \pi_0$ then $\theta \vDash \phi$ iff $\theta_{\geq 1} \vDash N_a^{-1}\phi$.*

Proof. For any formula ϕ, let L_ϕ be the statement: "for all structures, and paths θ through that structure, we have $\theta \vDash \phi$ iff $\theta_{\geq 1} \vDash N_a^{-1}\phi$."

It is clear that L_ϕ is true when ϕ is a state formula or a formula of the form $N\psi$. For some pair of formulae (ϕ, ψ), say that L_ϕ and L_ψ is true, then:
($\Longrightarrow$)

1. Say that $\theta \vDash (\phi U \psi)$,
   (a) If $\theta \vDash \psi$ then $\theta_{\geq 1} \vDash N_a^{-1}\psi$ and so $\theta_{\geq 1} \vDash N_a^{-1}(\phi U \psi)$.
   (b) If $\theta \nvDash \psi$ then $\theta \vDash \phi$ and $\theta_{\geq 1} \vDash (\phi U \psi)$. Hence $\theta_{\geq 1} \vDash N_a^{-1}(\psi)$ and so $\theta_{\geq 1} \vDash N_a^{-1}(\phi U \psi)$.

2. Say that $\theta \vDash \neg\phi$. Then $\theta \nvDash \phi$ and so $\theta_{\geq 1} \nvDash N_a^{-1}(\phi)$. Finally, $\theta_{\geq 1} \vDash \neg N_a^{-1}(\phi)$.
3. Say that $\theta \vDash \phi \wedge \psi$. Then $\theta \vDash \phi$ and $\theta \vDash \psi$. Thus $\theta_{\geq 1} \vDash N_a^{-1}\phi$ and $\theta_{\geq 1} \vDash N_a^{-1}\psi$. Hence $\theta_{\geq 1} \vDash N_a^{-1}(\phi) \wedge N_a^{-1}(\psi) = N_a^{-1}(\phi \wedge \psi)$
4. Say that $\theta \vDash \blacktriangle\phi$;
   (a) Thus $\sigma \vDash \phi$ for any deviation σ from θ. Note that as a deviation, $\sigma_0 = \theta_0$, and hence $\sigma_{\geq 1} \vDash N_a^{-1}(\phi)$; additionally for any path σ with $\sigma_0 = \theta_0$, if $\sigma_{\geq 1}$ is failure-free then σ is a deviation from θ and so $\theta \vDash ANON_a^{-1}(\phi)$. As Ξ is a normalization function, it follows that $\theta \vDash ANO\Xi\left(N_a^{-1}(\phi)\right)$.
   (b) We will show that $\theta_{\geq 1} \vDash \blacktriangle\Xi\left(N_a^{-1}(\phi)\right)$. Say that σ' is a deviation from $\theta_{\geq 1}$. Then from fusion closure of the set of paths B, there exists a path σ such that $\sigma_{\geq 1} = \sigma'$ and $\sigma_0 = \theta_0$. This path σ is a deviation from θ, and so $\sigma \vDash \phi$ and thus $\sigma' \vDash N_a^{-1}(\phi)$. Hence $\theta_{\geq 1} \vDash \blacktriangle N_a^{-1}(\phi)$.

($\Longleftarrow$)

1. Say that $\theta_{\geq 1} \vDash N_a^{-1}(\phi U\psi) = \left(N_a^{-1}(\phi) \wedge (\phi U\psi)\right) \vee N_a^{-1}(\psi)$
   (a) If $\theta_{\geq 1} \vDash N_a^{-1}(\phi) \wedge (\phi U\psi)$ then $\theta \vDash \phi$ and $\theta_{\geq 1} \vDash (\phi U\psi)$ so $\theta \vDash (\phi U\psi)$.
   (b) If $\theta_{\geq 1} \vDash N_a^{-1}(\psi)$ then $\theta \vDash \psi$ and so $\theta \vDash (\phi U\psi)$.
2. Say that $\theta_{\geq 1} \vDash \neg N_a^{-1}(\phi)$. Then $\theta_{\geq 1} \nvDash N_a^{-1}(\phi)$ and so $\theta \nvDash \phi$. Thus $\theta \vDash \neg\phi$.
3. Say that $\theta_{\geq 1} \vDash N_a^{-1}(\phi \wedge \psi)$. Then, from the definition of N_a^{-1} we have $\theta_{\geq 1} \vDash N_a^{-1}\phi$ and $\theta_{\geq 1} \vDash N_a^{-1}\psi$. Thus $\theta \vDash \phi$ and $\theta \vDash \psi$. Hence $\theta \vDash \phi \wedge \psi$
4. Say that $\theta_{\geq 1} \vDash N_a^{-1}(\blacktriangle\phi)$. Clearly $\theta_{\geq 1} \nvDash \bot$, and so $N_a^{-1}(\blacktriangle\phi) \neq \bot$. Thus, from Definition 16, we know that $N_a^{-1}(\blacktriangle\phi) = \blacktriangle N_a^{-1}(\phi)$ and that $ANO\left(N_a^{-1}(\phi)\right) \in a = h(\pi)$. It follows that $\theta \vDash ANO\left(N_a^{-1}(\phi)\right)$. Say σ is a deviation from θ; we will show that $\sigma_{\geq 1} \vDash N_a^{-1}(\phi)$, and so $\sigma \vDash \phi$. Since every deviation forces ϕ it follows that $\theta \vDash \blacktriangle\phi$:
   (a) as $\theta \vDash ANO\left(N_a^{-1}(\phi)\right)$, if σ is a 0-deviation then $\sigma_{\geq 1} \vDash N_a^{-1}(\phi)$;
   (b) if σ is an i-deviation where $i > 0$, then $\sigma_{\leq 1} = \theta_{\leq 1}$ and so $\sigma_{\geq 1}$ is an $(i-1)$-deviation from $\theta_{\geq 1}$. As $\theta_{\geq 1} \vDash \blacktriangle N_a^{-1}(\phi)$, it follows that $\sigma_{\geq 1} \vDash N_a^{-1}(\phi)$.

By induction on the length of the formula we see that the lemma holds.

Definition 18 (r_X). *The temporal successor r_X relation on hues below is defined as in Reynolds [12], but with the additional requirements (R5) and (R6); For all hues a, b put (a, b) in r_X iff the following conditions are satisfied:*

(R1) $N\alpha \in a$ implies $\alpha \in b$
(R2) $\neg N\alpha \in a$ implies $\alpha \notin b$
(R3) $\alpha U\beta \in a$ and $\beta \notin a$ implies $\alpha U\beta \in b$
(R4) $\neg(\alpha U\beta) \in a$ and $\alpha \in a$ implies $\neg(\alpha U\beta) \in b$
(R5) $\blacktriangle\alpha \in a$ implies $N_a^{-1}(\blacktriangle\alpha) \in b$
(R6) $\neg\blacktriangle\alpha \in a$ implies $\neg N_a^{-1}(\blacktriangle\alpha) \in b$

Definition 19 (r_A). *For hues a, b, we put (a, b) in r_A iff the following conditions hold:*

(A1) $A\alpha \in a$ iff $A\alpha \in b$ and $O\alpha \in a$ iff $O\alpha \in b$
(A2) For all $p \in \mathcal{V}$, we have $p \in a$ iff $p \in b$

Note that if $(a,b) \in r_A$ then for all formulae ϕ we have $N_a^{-1}(\phi) = N_b^{-1}(\phi)$ (i.e. $N_a^{-1} = N_b^{-1}$). The r_A relation is used to specify which pairs of hues can exist in the same "colour"; a colour represents a set of hues for paths which could start at the same state.

Definition 20. *A set of hues C is a colour of ϕ iff*

(C1) for all $a, b \in C$ we have $(a,b) \in r_A$; and
(C2) if $a \in C$ and $\neg A\alpha \in a$ or $\neg\blacktriangle\alpha \in a$ then there is $b \in C$ such that $\neg\alpha \in b$; and
(C3) if $a \in C$ and $\neg O\alpha \in a$ then there is $b \in C$ such that $\neg\alpha \in b$ and $\gamma \in b$; and
(C4) there exists $a \in C$ such that $\gamma \in a$

Let C_ϕ be the colours of ϕ. We define a successor relation on C_ϕ as follows:.

Definition 21 (R_X). *We define a temporal successor function R_X on colours as follows: for all $C, D \in C_\phi$, put $(C,D) \in R_X$ iff for all $b \in D$ there exists $a \in C$ such that $(a,b) \in r_X$.*

4 Pruning the Tableau

To adapt the pruning technique in [12] to RoBCTL* we add a new type of eventuality: $\neg\blacktriangle\psi$. This formula can be interpreted as "either $\neg\psi$ or there exists a path which eventually deviates, and $\neg\psi$ holds along that path." It is necessary to handle this eventuality. Imagine a tableau with only one colour $C = \{\{\neg\blacktriangle Gp, Gp, p, \neg\mathrm{Viol}, \top\}\}$, it is clear that $(C,C) \in R_X$ so without handling eventualities of the form $\neg\blacktriangle\psi$ this tableau would be accepted. We need to ensure that eventually a path deviates on which Gp is false.

Initially, we let the set S' of colours equal C_ϕ. We say that a 3-tuple (C,c,α) is an instance iff $C \in S'$, c is a hue, α is a formula and $\alpha \in c \in C$. We iteratively remove colours from S' according to the following rules until no more colours can be removed:

1. Remove C from S' if we cannot find successors for every hue in C. That is we remove C from S' if there exists a hue c in C such that for every $D \in S'$,
   (a) $(C,D) \notin R_X$, or
   (b) for every $d \in D$, the pair $(c,d) \notin r_X$.
2. An instance $(C,c,\alpha U\beta)$ is directly fulfilled iff $\beta \in c$. Initially, an instance is fulfilled iff it is directly fulfilled; we iteratively mark $(C,c,\alpha U\beta)$ as fulfilled iff there exists a fulfilled instance $(D,d,\alpha U\beta)$ such that $(C,D) \in R_X$ and $(c,d) \in r_X$. We finish when we can no longer mark instances as fulfilled. Finally, for all instances $(C,c,\alpha U\beta)$ that are not fulfilled, we remove C from S'.

3. An instance $(C, c, \neg\blacktriangle\alpha)$ is directly fulfilled iff $ANO\Xi\left(N_a^{-1}(\phi)\right) \notin c$ or $\alpha \notin c$. Initially, an instance is fulfilled iff it is directly fulfilled; we iteratively mark $(C, c, \neg\blacktriangle\alpha)$ as fulfilled iff there exists a fulfilled instance $(D, d, \neg\blacktriangle\alpha')$ such that $(C, D) \in R_X$, $(c, d) \in r_X$ and $\blacktriangle\alpha' = N_c^{-1}(\blacktriangle\alpha)$; we finish when we can no longer mark instances as fulfilled. Finally, for all instances $(C, c, \neg\blacktriangle\alpha)$ that are not fulfilled, we remove C from S'.

We say that the tableau succeeds if there exists a hue h and colour C such that $\phi \in h \in C \in S'$.

5 Cardinality of the Closure Set

The complexity of the tableau is doubly exponential [12] with respect to $|\mathbf{cl}\phi|$. We see that $|\mathbf{cl}\phi|$ is linear with respect to $|\phi| \max_{\psi \leq \phi} |N^\star(\psi)|$, from Definition 13 of $\mathbf{cl}\phi$. Thus if $|N^\star(\psi)|$ is n-exponential then the overall complexity of the tableau is $(n+2)$-exponential. In this section we will discuss the size of $|N^\star(\psi)|$.

Theorem 1. *For any formula of finite length, $|N^\star(\psi)|$ is finite and is $3m$-exponential on the size of the formulae when we require that there are no more than m pairs of alternations between $\blacktriangle$ and U that are not broken by an A (or O).*

We prove this by showing that we can build $|N^\star(\psi)|$ recursively: below we show that we can recurse through any number of $\blacktriangle$ operators with singly exponential blowup until we reach a U operator, and we can recurse through any number of U operators with doubly exponential blowup until we reach a $\blacktriangle$ operator;

Lemma 2. *Say that Φ is a set of array of formulae, and ψ is a formula constructed from any number of instances of $\wedge$ and $\neg$ operators, x instances of state formulae, y instances of $\blacktriangle$ operators, and z instances of N operators (we exclude the U operator) and elements of Φ. Let $\#(\phi)$, be the number of times that ϕ occurs in ψ without being part of a state formula. Then $|N^\star(\psi)|$ is singly exponential with respect to $\max_{\phi\in\Phi} N^\star(\phi)$.*

Proof. Consider the set $N^{-i}(\psi)$. By inspecting the definition of N^{-1} we see that we have two choices when we reach a state formula, $\top$ and $\bot$. When we reach a $\blacktriangle$ operator, we have two choices, terminate with $\bot$ or continue to recurse. It is clear that for all $\phi \in \Phi$ and $j \in [0, \infty]$ it is the case that $\left|N^{-j}(\phi)\right| \leq |N^\star(\phi)|$. It follows that:

$$\left|N^{-i}(\psi)\right| \leq 2^x 2^y \prod_{\phi\in\Phi} |N^\star(\phi)|^{\#(\phi)}$$

Note that for $i > z$ we have already removed all N in ψ that do not form part of an element of $|N^\star(\psi)|$, and have already replaced all state formulae do not form part of an element of $|N^\star(\psi)|$ with either $\top$ or $\bot$. It follows that

$$\left|\bigcup_{i>z} N^{-i}(\psi)\right| \leq 2^x 2^y \prod_{\phi\in\Phi} |N^\star(\phi)|^{\#(\phi)}.$$

Since $N^\star = \bigcup_{i\geq 0} \Xi N^{-i}(\psi)$ and $N^{-0}(\psi) = 1$,

$$|N^\star(\psi)| \leq \left|\bigcup_{i>0} N^{-i}(\psi)\right| \leq 1 + (1+z)\,2^x 2^y \prod_{\phi\in\Phi} |N^\star(\phi)|^{\#(\phi)}.$$

Lemma 3. *Say that Φ is a set of formulae each starting with $\blacktriangle$, and ψ is a formula constructed from x instances of state formulae, y instances of $\wedge, U, N, \neg$ operators (we exclude the $\blacktriangle$ operator) and elements of Φ. Let $\#(\phi)$, be the number of times that ϕ occurs in ψ without being part of a state formula. Then $N^\star(\phi)$ is doubly exponential with respect to* $\max_{\phi\in\Phi} N^\star(\phi)$.

Proof. Let $\mathbf{C}$ be a function such that $\mathbf{C}(\Phi)$ represents all normalized classical formulae with the elements of Φ as atoms. Note that a truth table on n atoms has 2^n rows, and hence there are 2^{2^n} equivalence classes on such formulae. It follows that $|\mathbf{C}(\Phi)| \leq 2^{2^{|\Phi|}}$. Let $\oint$ be a function from formulae to sets of formulae such that $\phi \in \oint(\psi)$ iff $N\phi \leq \psi$ or $\phi = (\phi_1 U \phi_2) \leq \psi$. For any set of formulae Φ, we define $\oint(\Phi)$ as $\bigcup_{\phi\in\Phi} \oint(\phi)$. We see that the following statement holds:

$$N^\star(\psi) \subseteq \mathbf{C}\left(= \oint(\psi) \cup \left(\bigcup_{\blacktriangle\phi\in\Phi} N^\star(\blacktriangle\phi)\right)\right)$$

It follows that $N^\star(\psi) \in \mathcal{O}\left(2^{2^{(x+y)+\sum_{\blacktriangle\phi\in\Phi} N^\star(\blacktriangle\phi)}}\right)$. In other words, $N^\star(\phi)$ is doubly exponential with respect to $\max_{\phi\in\Phi} N^\star(\phi)$.

Definition 22. *Let $\blacktriangle^n$ be a sequence $\blacktriangle\blacktriangle\ldots\blacktriangle$ of n instances of the $\blacktriangle$ operator.*

We see that also $N^\star(\blacktriangle^n\phi) = \{\blacktriangle x : x \in N^\star(\phi)\} \cup \{\bot\}$. The $\blacktriangle^n$ shorthand operator is interesting as it represents the statement "even with n additional failures" and a significant factor in the design of the $\blacktriangle$ was to provide a simple unimodal operator that could represent this statement. However, in the QCTL* based decision procedure for RoCTL* [7] the $\blacktriangle^n$ operator involves a non-elementary blowup in the complexity. By comparison, in this tableau the complexity is independent of n in $\blacktriangle^n$.

6 Soundness and Completeness

The proof of soundness and completeness [10] is too long to be reproduced here. It broadly follows the proof for BCTL*. However a substantial amount of extra material is required to handle the eventualities of the form $\neg\blacktriangle\psi$. Note that while eventualities of the form $(\alpha U \beta)$ remain unchanged until they are resolved by β occurring, eventualities of the form $\neg\blacktriangle\psi$ change over time. For example, the eventuality $\neg\blacktriangle N\phi$ becomes $\neg\blacktriangle\phi$ at the next step.

7 Conclusion

We have presented a tableau for RoBCTL*. This tableau is more efficient than the QCTL* based decision procedure for RoCTL*. The QCTL* based decision procedure has an exponential blowup for every alternation of propositional quantifiers [6,5] whereas this tableau only has exponential blowup on alternations between ▲ and U that are not broken by an A or O. In the examples of RoCTL* formulae presented previously [7], each clause either has the U or ▲ nested directly within an A or O, so it seems that most use cases for RoBCTL* would only require subclasses of RoBCTL* that are elementary to decide. Additionally, if a formula has no U operators, then as well as being elementary to decide, the interpretation of the formulae is the same in RoBCTL* as RoCTL*. Note also that the blowup only applies to the subformulae contained within the nesting, so systems represented by the conjunction of many clauses each of short length can be reasoned with efficiently; the examples [11,7] demonstrating the expressivity of the RoCTL* follow this pattern.

We hope that a truly elementary decision procedure for Ro(B)CTL* will be found. However, the degenerate cases that are inefficient on this decision procedure give a clue as to where a proof that RoCTL* is non-elementary may be found. The RoCTL logic [3] includes an ▲G operator. It would be interesting to discover if, unlike this tableau, the resolution procedure [3] for RoCTL was efficient for nestings of ▲G. Also of interest is that alternations between ▲ and Δ do not increase the complexity of the tableau based decision procedure.

Although finding an elementary decision procedure for RoBCTL* or RoCTL* is important, optimising for actual formulae would be of more practical interest. Even if it were to be shown that RoCTL* was 2-exponential like CTL*, this would not necessarily mean that it would be feasible to decide even small example formulae. Even the simple co-ordinated attack problem [7] required over 50 symbols to specify. Thus research into the actual time taken of a sample implementation on example formulae would be of interest. For many purposes the RoCTL fragment [3] of RoCTL* may be sufficient, and RoCTL may be easier to decide than CTL* or BCTL*.

Now having decision procedures for RoCTL*-like logics, we intend to apply RoCTL* to further practical problems. Some examples of how RoCTL* can succinctly represent robustness properties of simple systems have been given [7]. However, to test the decision procedures fully much larger and more complex examples need to be formalised and examined. For some uses, RoCTL* may need to be extended. RoCTL* can use the prone operator to discuss whether it is possible for a failure to be detected at a particular step. Diagnosis problems require that failures *will* be detected. For these purposes, an "If *at least* one additional failure occurs" operator and a knowledge operator are desirable. It would be useful to find an extension that satisfies these requirements while preserving decidability. Since the tableau does not rely on hiding the Viol atom, adding such an operator to the tableau should be trivial.

References

1. Aldewereld, H., Grossi, D., Vazquez-Salceda, J., Dignum, F.: Designing normative behaviour by the use of landmarks. In: Agents, Norms and Institutions for Regulated Multiag. Syst., Utrecht, The Netherlands (July 2005)
2. Broersen, J., Dignum, F., Dignum, V., Meyer, J.-J.Ch.: Designing a Deontic Logic of Deadlines. In: Lomuscio, A., Nute, D. (eds.) DEON 2004. LNCS (LNAI), vol. 3065, pp. 43–56. Springer, Heidelberg (2004)
3. Dixon, C., Mc Cabe-Dansted, J.C.: Resolution for a temporal logic of robustness (extended version). Technical Report ULCS-08-002, University of Liverpool, Department of Computer Science (2008), http://www.csc.liv.ac.uk/research/techreports/
4. Allen Emerson, E., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. In: STOC, pp. 169–180. ACM, New York (1982)
5. Emerson, E.A., Sistla, A.P.: Deciding branching time logic. In: STOC 1984: Proc. 16th annual ACM sympos. on Theory of computing, pp. 14–24. ACM Press, New York (1984)
6. French, T.: Decidability of quantifed propositional branching time logics. In: AI 2001: Proc. 14th Australian Joint Conf. on AI, London, UK, pp. 165–176. Springer, Heidelberg (2001)
7. French, T., Mc Cabe-Dansted, J.C., Reynolds, M.: A Temporal Logic of Robustness. In: Konev, B., Wolter, F. (eds.) FroCos 2007. LNCS (LNAI), vol. 4720, pp. 193–205. Springer, Heidelberg (2007); expanded tech. report [11], http://dx.doi.org/10.1007/978-3-540-74621-8_13
8. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
9. Long, W., Sato, Y., Horigome, M.: Quantification of sequential failure logic for fault tree analysis. Reliab. Eng. Syst. Safe. 67, 269–274 (2000)
10. Mc Cabe-Dansted, J.C.: A tableau for RoBCTL*. Technical report, UWA (2008), http://www.csse.uwa.edu.au/~john/papers/Da08Tableau.pdf
11. Mc Cabe-Dansted, J.C., French, T., Reynolds, M.: A temporal logic of robustness, RoCTL*. Technical report, UWA (2007), http://www.csse.uwa.edu.au/~john/papers/RoCTL07.pdf
12. Reynolds, M.: A Tableau for Bundled CTL. Journal of Logic and Computation 17(1), 117–132 (2007)
13. Rodrigo, A., Eduardo, A.: Normative pragmatics for agent communication languages. In: Akoka, J., Liddle, S.W., Song, I.-Y., Bertolotto, M., Comyn-Wattiau, I., van den Heuvel, W.-J., Kolp, M., Trujillo, J., Kop, C., Mayr, H.C. (eds.) ER Workshops 2005. LNCS, vol. 3770, pp. 172–181. Springer, Heidelberg (2005)

A Proof-Theoretic Approach to Deciding Subsumption and Computing Least Common Subsumer in $\mathcal{EL}$ w.r.t. Hybrid TBoxes

Novak Novaković

Theoretical Computer Science, TU Dresden, Germany

Abstract. Hybrid $\mathcal{EL}$-TBoxes combine general concept inclusions (GCIs), which are interpreted with descriptive semantics, with cyclic concept definitions, which are interpreted with greatest fixpoint (gfp) semantics. We introduce a proof-theoretic approach that yields a polynomial-time decision procedure for subsumption, and present a proof-theoretic computation of least common subsumers in $\mathcal{EL}$ w.r.t. hybrid TBoxes.

1 Introduction

The $\mathcal{EL}$-family of description logics (DLs) is a family of inexpressive DLs whose main distinguishing feature is that they provide their users with existential restrictions rather than value restrictions as the main concept constructor involving roles. The core language of this family is $\mathcal{EL}$, which has the top concept ($\top$), conjunction ($\sqcap$), and existential restrictions ($\exists r.C$) as concept constructors. This family has recently drawn considerable attention since, on the one hand, the subsumption problem stays tractable (i.e., decidable in polynomial time) in situations where the corresponding DL with value restrictions becomes intractable. In particular, subsumption in $\mathcal{EL}$ is tractable both w.r.t. cyclic TBoxes interpreted with gfp or descriptive semantics [3] and w.r.t. general TBoxes (i.e., finite sets of GCIs) interpreted with descriptive semantics [6,4]. On the other hand, although of limited expressive power, $\mathcal{EL}$ is nevertheless used in applications, e.g., to define biomedical ontologies. For example, both the large medical ontology SNOMED CT [14] and the Gene Ontology [1] can be expressed in $\mathcal{EL}$, and the same is true for large parts of the medical ontology GALEN [12].

In some cases, it would be advantageous to have both GCIs interpreted with descriptive semantics and cyclic concept definitions interpreted with gfp-semantics available in one TBox. One motivation for such hybrid TBoxes comes from the area of non-standard inferences in DLs. For example, if one wants to support the so-called bottom-up construction of DL knowledge bases, then one needs to compute least common subsumers (lcs) and most specific concepts (msc) [5]. In [2], it was shown that the lcs and the msc in $\mathcal{EL}$ always exist and can be computed in polynomial time if cyclic definitions that are interpreted with gfp-semantics are available. In contrast, if cyclic definitions or GCIs are interpreted

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 311–323, 2008.

with descriptive semantics, neither the lcs nor the msc need to exist. Hybrid $\mathcal{EL}$-TBoxes have first been introduced in [8]. Basically, such a TBox consists of two parts $\mathcal{T}$ and $\mathcal{F}$, where $\mathcal{T}$ is a cyclic TBox whose primitive concepts occur in the GCIs of the general TBox $\mathcal{F}$. However, defined concepts of $\mathcal{T}$ must not occur in $\mathcal{F}$. It was shown in [8] that subsumption w.r.t. such hybrid TBoxes can still be decided in polynomial time. The algorithm uses reasoning w.r.t. the general TBox $\mathcal{F}$ to extend the cyclic TBox $\mathcal{T}$ to a cyclic TBox $\widehat{\mathcal{T}}$ such that subsumption can then be decided considering only $\widehat{\mathcal{T}}$. In [7] it was shown that, w.r.t. hybrid $\mathcal{EL}$-TBoxes, the lcs and msc always exists and can be computed in polynomial time.

Both, the existing algorithm for deciding subsumption ([8]), and the algorithms for computing lcs and mcs ([7]) in $\mathcal{EL}$ w.r.t. hybrid TBoxes include a pre-processing step of *normalization* of the terminologies. Normalization is, consequently, also required for the algorithms for deciding subsumption, and the algorithms for computing lcs and mcs in $\mathcal{EL}$ w.r.t. descriptive and greatest fixpoint semantics from [13]. This pre-processing step has two undesirable features. From the complexity point of view, it causes quadratic blow-up of the terminologies, and thus, a quadratic blowup in the size of the input to the algorithms. Even more important, especially in the cases of lcs and mcs, normalization replaces the original concept definitions from the terminologies by new ones, by introducing new concept names that occur in those modified definitions. For instance, assume one wants to extend an existing large life-science ontology (and those are usually not normalized) by adding just a single lcs of some two defined concepts. The existing procedure results in quadratic blow-up of the entire ontology, its modification for all users of the ontology, and introduction of some new (generic and unintuitive) concept names.

An approach for deciding subsumption in $\mathcal{EL}$ that significantly differs from the ones described in [3,6,4] was introduced in [9]. It is based on sound and complete Gentzen-style proof calculi for subsumption w.r.t. cyclic TBoxes interpreted with gfp semantics and for subsumption w.r.t. general TBoxes interpreted with descriptive semantics. These calculi yield polynomial-time decision procedures since they satisfy an appropriate sub-description property.

This paper shows that a polynomial-time decision procedure can be obtained for deciding subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes by combining the two calculi introduced in [9]. Another contribution of this paper is a proof-theoretic computation of lcs in $\mathcal{EL}$ w.r.t. hybrid TBoxes. In both cases, the normalization of the ontologies is avoided, together with the undesirable features that come along with it.

2 Hybrid $\mathcal{EL}$-TBoxes

Starting with a set N_{con} of concept names and a set N_{role} of role names, $\mathcal{EL}$-*concept descriptions* are built using the concept constructors top concept ($\top$), conjunction ($\sqcap$), and existential restrictions ($\exists r.C$). The semantics of $\mathcal{EL}$ is defined in the usual way, using the notion of an interpretation $\mathcal{I} = (\mathcal{D}_\mathcal{I}, \cdot^\mathcal{I})$, which consists of a nonempty domain $\mathcal{D}_\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$ that assigns

Table 1. Syntax and semantics of $\mathcal{EL}$

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \mathcal{D}_{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \mathcal{D}_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}}$
top-concept	$\top$	$\top^{\mathcal{I}} = \mathcal{D}_{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
exist. restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
subsumption	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

binary relations on $\mathcal{D}_{\mathcal{I}}$ to role names and subsets of $\mathcal{D}_{\mathcal{I}}$ to concept descriptions, as shown in the semantics column of Table 1. A *concept definition* is an expression of the form $A \equiv C$, where A is a concept name and C is a concept description, and a *general concept inclusion* (GCI) is an expression of the form $C \sqsubseteq D$, where C, D are concept descriptions. An interpretation $\mathcal{I}$ is a *model* of a concept definition or GCI if it satisfies the respective condition given in the semantics column of Table 1. This semantics for GCIs and concept definitions is usually called *descriptive semantics*. A *TBox* is a finite set $\mathcal{T}$ of concept definitions that does not contain multiple definitions, i.e., $\{A \equiv C, A \equiv D\} \subseteq \mathcal{T}$ implies $C = D$. Note that TBoxes are *not* required to be *acyclic*, i.e., there may be cyclic dependencies among the concept definitions. A *general TBox* is a finite set of GCIs. The interpretation $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$ (the general TBox $\mathcal{F}$) iff it is a model of all concept definitions (GCIs) in $\mathcal{T}$ (in $\mathcal{F}$). The name *general* TBox is justified by the fact that concept definitions $A \equiv C$ can of course be expressed by GCIs $A \sqsubseteq C, C \sqsubseteq A$. However, in hybrid TBoxes to be considered, concept definitions will be interpreted by greatest fixpoint semantics rather than by descriptive semantics. We assume in the following that the set of concept names N_{con} is partitioned into the set of *primitive concept names* N_{prim} and the set of *defined concept names* N_{def}. In a hybrid TBox, concept names occurring on the left-hand side of a concept definition are required to come from the set N_{def}, whereas GCIs may not contain concept names from N_{def}.

Definition 1 (Hybrid $\mathcal{EL}$-TBoxes). *A* hybrid $\mathcal{EL}$-TBox *is a pair* $(\mathcal{F}, \mathcal{T})$, *where* $\mathcal{F}$ *is a general* $\mathcal{EL}$*-TBox containing only concept names from* N_{prim}, *and* $\mathcal{T}$ *is an* $\mathcal{EL}$*-TBox such that* $A \equiv C \in \mathcal{T}$ *implies* $A \in N_{def}$.

An example of a hybrid $\mathcal{EL}$-Tbox, taken from [8], is given in Fig. 1. It defines the concepts 'disease of the connective tissue,' 'bacterial infection,' and 'bacterial pericarditis' using the cyclic definitions in $\mathcal{T}$. The general TBox $\mathcal{F}$ states some properties that the primitive concepts and roles occurring in $\mathcal{T}$ must satisfy, such as the fact that a disease located on connective tissue also acts on the connective tissue. In general, the idea underlying the definition of hybrid TBoxes is the following: $\mathcal{F}$ can be used to constrain the interpretation of the primitive concepts and roles,

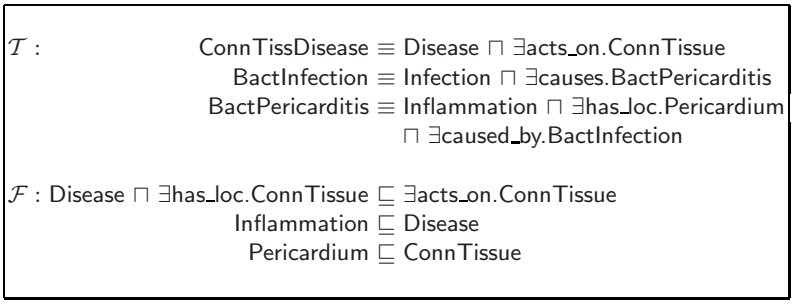


Fig. 1. A small hybrid $\mathcal{EL}$-TBox

whereas $\mathcal{T}$ tells us how to interpret the defined concepts occurring in it, once the interpretation of the primitive concepts and roles is fixed.

A *primitive interpretation* $\mathcal{J}$ is defined like an interpretation, with the only difference that it does not provide an interpretation for defined concepts. A primitive interpretation can thus interpret concept descriptions built over N_{prim} and N_{role}, but it cannot interpret concept descriptions containing elements of N_{def}. Given a primitive interpretation $\mathcal{J}$, we say that the (full) interpretation $\mathcal{I}$ is *based on* $\mathcal{J}$ if it has the same domain as $\mathcal{J}$ and its interpretation function coincides with $\mathcal{J}$ on N_{prim} and N_{role}.

Given two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ based on the same primitive interpretation $\mathcal{J}$, we define

$$\mathcal{I}_1 \preceq_{\mathcal{J}} \mathcal{I}_2 \quad \text{, iff} \quad A^{\mathcal{I}_1} \subseteq A^{\mathcal{I}_2} \text{ for all } A \in N_{def}.$$

It is easy to see that the relation $\preceq_{\mathcal{J}}$ is a partial order on the set of interpretations based on $\mathcal{J}$. In [3] the following was shown: given an $\mathcal{EL}$-TBox $\mathcal{T}$ and a primitive interpretation $\mathcal{J}$, there exists a unique model $\mathcal{I}$ of $\mathcal{T}$ such that

- $\mathcal{I}$ is based on $\mathcal{J}$;
- $\mathcal{I}' \preceq_{\mathcal{J}} \mathcal{I}$ for all models $\mathcal{I}'$ of $\mathcal{T}$ that are based on $\mathcal{J}$.

We call such a model $\mathcal{I}$ a *gfp-model* of $\mathcal{T}$.

Definition 2 (Semantics of hybrid $\mathcal{EL}$-TBoxes). *An interpretation $\mathcal{I}$ is a* hybrid model *of the hybrid $\mathcal{EL}$-TBox $(\mathcal{F}, \mathcal{T})$, iff $\mathcal{I}$ is a gfp-model of $\mathcal{T}$ and the primitive interpretation $\mathcal{J}$ it is based on is a model of $\mathcal{F}$.*

It is well-known that gfp-semantics coincides with descriptive semantics for acyclic TBoxes. Thus, if $\mathcal{T}$ is actually acyclic, then $\mathcal{I}$ is a hybrid model of $(\mathcal{F}, \mathcal{T})$ according to the semantics introduced in Definition 2, iff it is a model of $\mathcal{T} \cup \mathcal{F}$ w.r.t. descriptive semantics, i.e., iff $\mathcal{I}$ is a model of every GCI in $\mathcal{F}$ and of every concept definition in $\mathcal{T}$.

3 Subsumption w.r.t. Hybrid $\mathcal{EL}$-TBoxes

Based on the semantics for hybrid TBoxes introduced above, we can now define the main inference problem that we want to solve in this paper.

Definition 3 (Subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes). *Let $(\mathcal{F},\mathcal{T})$ be a hybrid $\mathcal{EL}$-TBox, and A, B defined concepts occurring on the left-hand side of a definition in $\mathcal{T}$. Then A* is subsumed by *B* w.r.t. *$(\mathcal{F},\mathcal{T})$ (written $A \sqsubseteq_{gfp,\mathcal{F},\mathcal{T}} B$), iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for all hybrid models $\mathcal{I}$ of $(\mathcal{F},\mathcal{T})$.*

Defining (and computing) subsumption only for concept names A, B defined in $\mathcal{T}$ rather than for arbitrary concept descriptions C, D is not a real restriction since one can always add definitions with the right-hand sides C, D to $\mathcal{T}$.

Assume that the hybrid $\mathcal{EL}$-TBox $(\mathcal{F},\mathcal{T})$ is given, and that we want to decide whether, for given defined concepts A, B, the subsumption relationship $A \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}} B$ holds or not. Following the ideas in [9], we introduce a sound and complete Gentzen-style calculus for subsumption. The reason why this calculus yields a decision procedure is basically that it has the sub-description property, i.e., application of rules can be restricted to sub-descriptions of concept descriptions occurring in $\mathcal{F}$ or $\mathcal{T}$.

A *sequent for* $(\mathcal{F},\mathcal{T})$ is of the form $C \sqsubseteq_n D$, where C, D are sub-descriptions of concept descriptions occurring in $\mathcal{F}$ or $\mathcal{T}$, and $n \geq 0$. The rules of the **H**ybrid $\mathcal{EL}$-TBox **C**alculus HC depicted in Fig. 2 can be used to derive new sequents from sequents that have already been derived. For example, the sequents in the first row of the figure can always be derived without any prerequisites, using the rules Refl, Top, and Start, respectively. Using the rule AndR, the sequent $C \sqsubseteq_n D \sqcap E$ can be derived in case both $C \sqsubseteq_n D$ and $C \sqsubseteq_n E$ have already been derived. Note that the rule Start applies only for $n = 0$. Also note that, in the rule DefR, the index is incremented when going from the prerequisite to the consequent.

$$C \sqsubseteq_n C \quad \text{(Refl)} \qquad C \sqsubseteq_n \top \quad \text{(Top)} \qquad C \sqsubseteq_0 D \quad \text{(Start)}$$

$$\frac{C \sqsubseteq_n E}{C \sqcap D \sqsubseteq_n E} \quad \text{(AndL1)} \qquad \frac{D \sqsubseteq_n E}{C \sqcap D \sqsubseteq_n E} \quad \text{(AndL2)} \qquad \frac{C \sqsubseteq_n D \quad C \sqsubseteq_n E}{C \sqsubseteq_n D \sqcap E} \quad \text{(AndR)}$$

$$\frac{C \sqsubseteq_n D}{\exists r.C \sqsubseteq_n \exists r.D} \quad \text{(Ex)}$$

$$\frac{C \sqsubseteq_n D}{A \sqsubseteq_n D} \quad \text{(DefL)} \quad \text{for } A \equiv C \in \mathcal{T} \qquad \frac{D \sqsubseteq_n C}{D \sqsubseteq_{n+1} A} \quad \text{(DefR)} \quad \text{for } A \equiv C \in \mathcal{T} \qquad \frac{C \sqsubseteq_n E \quad F \sqsubseteq_n D}{C \sqsubseteq_n D} \quad \text{(GCI)} \quad \text{for } E \sqsubseteq F \in \mathcal{F}$$

Fig. 2. The rule system HC

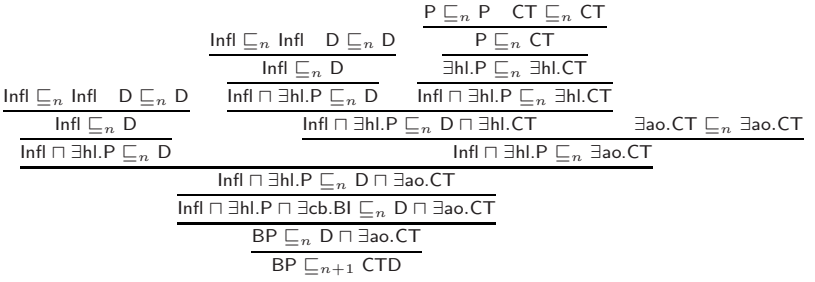


Fig. 3. An example of a derivation in HC

Fig. 3 shows a derivation in HC w.r.t. the hybrid $\mathcal{EL}$-TBox from Fig. 1, where obvious abbreviations of concept and role names have been made. This derivation tree demonstrates that the sequent $\mathsf{BactPericarditis} \sqsubseteq_{n+1} \mathsf{ConnTissDisease}$ can be derived for every $n \geq 0$. Note that we can also derive $\mathsf{BactPericarditis} \sqsubseteq_0 \mathsf{ConnTissDisease}$ using the rule Start.

The calculus HC defines binary relations $\sqsubseteq_n$ for $n \in \{0, 1, \ldots\} \cup \{\infty\}$ on the set of sub-descriptions of concept descriptions occurring in $\mathcal{F}$ or $\mathcal{T}$:

Definition 4. *Let C, D be sub-descriptions of the concept descriptions occurring in $\mathcal{F}$ or $\mathcal{T}$. Then $C \sqsubseteq_n D$ holds, iff the sequent $C \sqsubseteq_n D$ can be derived using the rules of HC. In addition, $C \sqsubseteq_\infty D$ holds, iff $C \sqsubseteq_n D$ holds for all $n \geq 0$.*

The calculus HC is sound and complete for subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes in the following sense.

Theorem 1 (Soundness and Completeness of HC). *Let $(\mathcal{F}, \mathcal{T})$ be a hybrid $\mathcal{EL}$-TBox, and A, B defined concepts occurring on the left-hand side of a definition in $\mathcal{T}$. Then $A \sqsubseteq_{gfp,\mathcal{F},\mathcal{T}} B$, iff $A \sqsubseteq_\infty B$ holds.*

A detailed proof of this theorem is given in [11]. Though the rules of HC are taken from the sound and complete subsumption calculi introduced in [9] for subsumption w.r.t. cyclic $\mathcal{EL}$-TBoxes interpreted with gfp-semantics and for subsumption w.r.t. general $\mathcal{EL}$-TBoxes interpreted with descriptive semantics, respectively, the proof that their combination is sound and complete for the case of hybrid $\mathcal{EL}$-TBoxes requires non-trivial modifications of the proofs given in [9]. Nevertheless, these proofs appear to be simpler and easier to comprehend than the ones given in [8,10] for the correctness of the reduction-based subsumption algorithm for hybrid $\mathcal{EL}$-TBoxes introduced there.

In our example, we have $\mathsf{BactPericarditis} \sqsubseteq_\infty \mathsf{ConnTissDisease}$, and thus soundness of HC implies that the subsumption relationship $\mathsf{BactPericarditis} \sqsubseteq_{\mathrm{gfp},\mathcal{F},\mathcal{T}} \mathsf{ConnTissDisease}$ holds.

It is not hard to show that $\sqsubseteq_0$ is the universal relation on sub-descriptions of the concept descriptions occurring in $\mathcal{F}$ or $\mathcal{T}$, and that $\sqsubseteq_{n+1} \subseteq \sqsubseteq_n$ holds for all $n \geq 0$ (see [11] for a proof). Thus, to compute $\sqsubseteq_\infty$ we can start with the universal relation $\sqsubseteq_0$, and then compute $\sqsubseteq_1, \sqsubseteq_2, \ldots$, until for some m we

have $\sqsubseteq_m = \sqsubseteq_{m+1}$, and thus $\sqsubseteq_m = \sqsubseteq_\infty$. Since the set of sub-descriptions is finite, the computation of each relation $\sqsubseteq_n$ can be done in finite time, and we can be sure that there always exists an m such that $\sqsubseteq_m = \sqsubseteq_{m+1}$. This shows that the calculus HC indeed yields a subsumption algorithm. Even more so, the decision procedure will terminate in polynomial time. This can easily be seen by noticing that we can compute each of the relations $\sqsubseteq_m$ in polynomial time by performing the proof search for each pair of subconcepts of the TBox with caching the intermediate derived subsumption pairs. Notice that the number of subconcepts of the TBox is linear in the size of the TBox. Also, the maximal number m_0 of different $\sqsubseteq_m$ relations corresponds to the case where $\sqsubseteq_i$ and $\sqsubseteq_{i+1}$ differ in a single subsumption pair, for all $i \leq m_0$. Thus, the total number of different $\sqsubseteq_m$ relations is bounded by the number of subsumption pairs $+1$, i.e. bounded by a quadratic function in the size of the TBox.

A detailed description of the implementation of this decision procedure can be found in [15].

4 Computing Least Common Subsumer in $\mathcal{EL}$ w.r.t. Hybrid TBoxes

This section is dedicated to employing the developed proof-theoretic techniques in calculating and showing the correctness of the computation of the least-common subsumer of two defined concepts with respect to hybrid TBoxes. We start by introducing the notion of a conservative extension of a hybrid $\mathcal{EL}$-TBox.

Definition 5. *Given a hybrid $\mathcal{EL}$-TBox $(\mathcal{F}, \mathcal{T}')$ we say that the hybrid TBox $(\mathcal{F}, \mathcal{T}'')$ is a* conservative extension *of $(\mathcal{F}, \mathcal{T}')$, iff $\mathcal{T}' \subseteq \mathcal{T}''$, and $\mathcal{T}'$ and $\mathcal{T}''$ have the same primitive concepts and roles.*

It is well known that the conservative extensions do not change the set of subsumption pairs (see [2]), i.e. $(\mathcal{F}, \mathcal{T}) \models C \sqsubseteq D$, iff $(\mathcal{F}, \mathcal{T}') \models C \sqsubseteq D$, for all subconcepts C, D occurring in $(\mathcal{F}, \mathcal{T})$. This can also be shown in a proof-theoretic way by noticing that $C \sqsubseteq_\infty D$ can be derived in the HC calculus for $(\mathcal{F}, \mathcal{T})$, iff it can be derived in HC for $(\mathcal{F}, \mathcal{T}')$.

Notice that, for instance, $\sqsubseteq_n$, $\sqsubseteq_\infty$ and N_{def} are defined w.r.t. a concrete TBox. That is why, in the cases where multiple TBoxes are concerned, superscripts are used to specify the appropriate TBoxes. The following definition introduces the notion of least-common subsumer in the hybrid setting.

Definition 6. *(Hybrid lcs) Let $(\mathcal{F}, \mathcal{T}_1)$ be a hybrid $\mathcal{EL}$-TBox and $A, B \in N_{def}^{\mathcal{T}_1}$. Let $(\mathcal{F}, \mathcal{T}_2)$ be a conservative extension of $(\mathcal{F}, \mathcal{T}_1)$ with $Z \in N_{def}^{\mathcal{T}_2}$. Then Z in $(\mathcal{F}, \mathcal{T}_2)$ is a hybrid least-common subsumer (lcs) of A, B in $(\mathcal{F}, \mathcal{T}_1)$, iff the following conditions hold:*

1. *$A \sqsubseteq_{gfp,\mathcal{F},\mathcal{T}_2} Z$ and $B \sqsubseteq_{gfp,\mathcal{F},\mathcal{T}_2} Z$; and*
2. *if $(\mathcal{F}, \mathcal{T}_3)$ is a conservative extension of $(\mathcal{F}, \mathcal{T}_2)$ and $D \in N_{def}^{\mathcal{T}_3}$ such that $A \sqsubseteq_{gfp,\mathcal{F},\mathcal{T}_3} D$ and $B \sqsubseteq_{gfp,\mathcal{F},\mathcal{T}_3} D$ then $Z \sqsubseteq_{gfp,\mathcal{F},\mathcal{T}_3} D$.*

Concept D from the previous definition is an arbitrary concept defined in some conservative extension $(\mathcal{F}, \mathcal{T}_3)$. It would suffice, though, to restrict D to be arbitrary concept defined in $\mathcal{T}_3 \setminus \mathcal{T}_1$, i.e. it is sufficient to consider only newly defined concepts for testing the condition 2 of the definition above. Indeed, if we want to test whether $Z \sqsubseteq_{gfp,\mathcal{F},\mathcal{T}_3} D$ for $D \in N_{def}^{\mathcal{T}_1}$, we can equivalently check whether $Z \sqsubseteq_{gfp,\mathcal{F},\mathcal{T}_4} A_D$, where $\mathcal{T}_4$ consists of $\mathcal{T}_2$ with a definition $D \equiv A_D$ for a new concept A_D.

One can observe that, as mentioned in the introduction, the existing algorithm for computing lcs for $\mathcal{EL}$ w.r.t. TBoxes interpreted by greatest fixpoint semantics from [13], and the algorithm for computing lcs for $\mathcal{EL}$ w.r.t. hybrid TBoxes from [7], strictly speaking, do not result in a conservative extension of the original TBox due to the normalization step.

Assume now that, given a hybrid TBox $(\mathcal{F}, \mathcal{T})$, one wants to know the least common subsumer of two defined concepts A and B occurring in a hybrid TBox. We give a definition of an extension of the hybrid TBox which contains definitions of lcs of defined concepts occurring in the original TBox.

Before doing so, consider the set *subcon* of all subconcepts of concept descriptions occurring in the TBox $(\mathcal{F}, \mathcal{T})$ and consider the sets

$$\begin{aligned} ExRest &= \{C \mid C \in subcon \text{ and there is an } r \in N_{role} \text{ such that } \exists r.C \in subcon\}, \\ N_{pair} &= \{(C, D) \mid C, D \in (N_{def}^{\mathcal{T}} \cup ExRest)\}, \text{ and} \\ Prims &= \{C \mid C \in subcon \text{ and } C \text{ does not have elments of } N_{def} \text{ as subconcepts}\}. \end{aligned}$$

Notice that elements of $Prims$ are concept descriptions built using only primitive concept names and role names. Now we define the conservative extension of the TBox as follows.

Definition 7. *Let $(\mathcal{F}, \mathcal{T})$ be a hybrid $\mathcal{EL}$-TBox. A conservative extension $(\mathcal{F}, \mathcal{T}_{lcs})$ of $(\mathcal{F}, \mathcal{T})$ is obtained by adding to the $(\mathcal{F}, \mathcal{T})$ definitions*

$$(C, D) \equiv \theta_1 \sqcap ... \sqcap \theta_k \sqcap X_1 \sqcap ... \sqcap X_l \sqcap \exists r_1.(C_1, D_1) \sqcap ... \sqcap \exists r_m.(C_m, D_m)$$

for each $(C, D) \in N_{pair}$, where:

1. *$\theta \in \{\theta_1, ..., \theta_k\}$, iff $C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T})} \theta$, $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T})} \theta$, and $\theta \in Prims$*
2. *$X \in \{X_1, ..., X_l\}$, iff $C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T})} X$, $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T})} X$, and $X \in N_{def}^{\mathcal{T}}$;*
3. *$\exists r.(\tau, \sigma) \in \{\exists r_1.(C_1, D_1), ..., \exists r_m.(C_m, D_m)\}$, iff $C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T})} \exists r.\tau$, $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T})} \exists r.\sigma$ and $(\tau, \sigma) \in N_{pair}$.*

Least common subsumer of two defined concepts A and B occurring in the TBox will be newly defined concept (A, B). Once the $\sqsubseteq_\infty^{(\mathcal{F},\mathcal{T})}$ relation is computed, computation of the extension $(\mathcal{F}, \mathcal{T}_{lcs})$ can be done by a simple computation in polynomial time, and the resulting TBox is indeed a conservative extension of the original one.

In what follows we show that the definition above provides computation of lcs of two given concepts, i.e. that the least common subsumer of two defined concepts A and B occurring in the TBox is the newly defined concept (A, B).

What needs to be shown is that conditions 1 and 2 from Definition 6 hold for all (A, B). In order to do so, we will simplify the discussion by restricting our attention to those conservative extensions from condition 2 for which newly added definitions are of a certain regular structure. Such an assumption, of course, is done without loss of generality.

Definition 8. *We say that a conservative extension $(\mathcal{F}, \mathcal{T}')$ of the hybrid $\mathcal{EL}$-TBox $(\mathcal{F}, \mathcal{T})$ is obtained by adding* normalized definitions modulo $(\mathcal{F}, \mathcal{T})$ *if every definition in $\mathcal{T}' \setminus \mathcal{T}$ is of the form:*

$$Z \equiv P_1 \sqcap ... \sqcap P_m \sqcap A_1 \sqcap ... \sqcap A_k \sqcap \exists r_1.B_1 \sqcap ... \sqcap \exists r_n.B_n$$

where P_i is a primitive concept for every $i = 1, ..., m$, A_i is a concept defined in $\mathcal{T}$, for every $i = 1, ..., k$, and B_j is a concept defined in $\mathcal{T}' \setminus \mathcal{T}$, for every $j = 1, ..., n$.

The proof of the following proposition can be found in [11].

Proposition 1. *Let $(\mathcal{F}, \mathcal{T})$ be a hybrid $\mathcal{EL}$-TBox, and $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_1)$ some conservative extension of $(\mathcal{F}, \mathcal{T})$. Then, there is a conservative extension $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_2)$ of $(\mathcal{F}, \mathcal{T})$ obtained by adding normalized definitions modulo $(\mathcal{F}, \mathcal{T})$ to it, such that the set of defined concepts in $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_1)$ is a subset of the set of defined concepts in $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_2)$, and $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_1) \models C \sqsubseteq D$, iff $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_2) \models C \sqsubseteq D$ for every two concepts C and D defined in $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_1)$.*

This proposition shows that one can restrict the attention to the conservative extensions obtained by adding the normalized definitions modulo a TBox when checking for property 2 from the definition of lcs. Indeed, let Φ be a concept defined in a conservative extension $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_1)$ of hybrid TBox $(\mathcal{F}, \mathcal{T})$, such that Φ subsumes both A and B. By the previous proposition, there is a conservative extension $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_2)$ of the TBox $(\mathcal{F}, \mathcal{T})$ by normalized definitions modulo $(\mathcal{F}, \mathcal{T})$, such that (A, B) will be subsumed by Φ w.r.t. $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_1)$, iff (A, B) is subsumed by Φ w.r.t. $(\mathcal{F}, \mathcal{T} \cup \mathcal{A}_2)$. In particular, (A, B) will be subsumed by every concept Φ that subsumes both A and B w.r.t. an arbitrary conservative extension of the TBox, iff it is subsumed by every concept Φ that subsume both A and B w.r.t. conservative extensions of the TBox obtained by adding normalized definitions modulo $(\mathcal{F}, \mathcal{T})$.

We continue by introducing a relation $\Vdash$. We say that $C \Vdash D$, iff $C \sqsubseteq_n D$ can be derived for some n using only the rules that consider the left-hand side of a sequent, i.e. (Ref), (AndL1), (AndL2), (DefL) and (GCI). Notice that if $C \sqsubseteq_n D$ can be derived using only those rules for some n, then it can be derived for any n. The following, rather technical lemmas are given here without proofs, which can be found in [11].

Lemma 1. *Suppose that $n > 0$.*

- *$F \sqsubseteq_{n+1} A$, iff $F \sqsubseteq_n C_A$, where $A \equiv C_A$ is an axiom of the TBox.*
- *$F \sqsubseteq_n \exists r.D$, iff there exist α, β, ρ such that $F \sqsubseteq_n \alpha, \beta \Vdash \exists r.\rho$ and $\rho \sqsubseteq_n D$ for some subconcept ρ of the TBox, and α and β being such that either $\alpha = \beta = F$ or $\alpha \sqsubseteq \beta$ being a GCI from $\mathcal{F}$.*

Lemma 2. *Let D, C and F be arbitrary subconcepts occurring in a TBox $(\mathcal{F},\mathcal{T})$. If $D \sqsubseteq_\infty F$ and $F \sqsubseteq_n C$, then $D \sqsubseteq_n C$.*

We are equipped now to show that our newly defined concepts are indeed the subsumers.

Lemma 3. *Let D and C be arbitrary concepts from $N_{def}^{\mathcal{T}} \cup ExRest$. Then, $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} (D,C)$ and $C \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} (D,C)$ for every n.*

Proof. We give a proof of $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} (D,C)$, proof of $C \sqsubseteq_n (D,C)$ is analogous. Proof is carried out by induction on n. For $n = 0$, $D \sqsubseteq_0^{(\mathcal{F},\mathcal{T}_{lcs})} (D,C)$ follows from the rule (Start). Assume now that $D \sqsubseteq_l^{(\mathcal{F},\mathcal{T}_{lcs})} (D,C)$ holds for all $l \leq n$. We prove that $D \sqsubseteq_{n+1}^{(\mathcal{F},\mathcal{T}_{lcs})} (D,C)$. Let

$$(D,C) \equiv \theta_1 \sqcap ... \sqcap \theta_k \sqcap X_1 \sqcap ... \sqcap X_u \sqcap \exists s_1.(D_{l_1}, C_{m_1}) \sqcap ... \sqcap \exists s_t.(D_{l_t}, C_{m_t})$$

be the definition of (D,C) in the extended hybrid TBox $(\mathcal{F},\mathcal{T}_{lcs})$. Lemma 1 applied to this definition yields $D \sqsubseteq_{n+1}^{(\mathcal{F},\mathcal{T}_{lcs})} (D,C)$, iff $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \theta_1 \sqcap ... \sqcap \theta_k \sqcap X_1 \sqcap ... \sqcap X_u \sqcap \exists s_1.(D_{l_1}, C_{m_1}) \sqcap ... \sqcap \exists s_t.(D_{l_t}, C_{m_t})$. Therefore, it is sufficient to show $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \theta_1 \sqcap ... \sqcap \theta_k \sqcap X_1 \sqcap ... \sqcap X_u \sqcap \exists s_1.(D_{l_1}, C_{m_1}) \sqcap ... \sqcap \exists s_t.(D_{l_t}, C_{m_t})$. Due to (AndR), one way to show this is to give a proof of $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \theta_i$ for $i = 1, ..., k$, $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} X_i$ for $i = 1, ..., u$, and $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \exists s_j.(D_{l_j}, C_{m_j})$ for $j = 1, ..., t$.

- $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \theta_i$: by Definition 7, $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_{lcs})} \theta_i$. Therefore, by definition of $\sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_{lcs})}$, $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \theta_i$, for $i = 1, ..., k$. Similarly, $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} X_i$.
- $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \exists s_j.(D_{l_j}, C_{m_j})$: by Definition 7, $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_{lcs})} \exists s_j.D_{l_j}$, therefore $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \exists s_j.D_{l_j}$. Since both D_{l_j} and C_{m_j} belong to the $N_{def} \cup ExRest$, induction hypothesis can be applied and it yields $D_{l_j} \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} (D_{l_j}, C_{m_j})$. Then, $\exists s_j.D_{l_j} \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \exists s_j.(D_{l_j}, C_{m_j})$ follows by applying the rule (Ex). Since $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_{lcs})} \exists s_j.D_{l_j}$, Lemma 2 yields $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})} \exists s_j.(D_{l_j}, C_{m_j})$.

By definition of $\sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_{lcs})}$, and due to the soundness of HC, both A and B are subsumed by (A,B), for all defined concepts A and B in $(\mathcal{F},\mathcal{T})$.

We give here another technical property of conservative extensions and $\Vdash^{(\mathcal{F},\mathcal{T})}$ relation.

Lemma 4. *Let $(\mathcal{F},\mathcal{T}_2)$ be an arbitrary conservative extension of $(\mathcal{F},\mathcal{T})$. If σ is a subconcept occurring in $(\mathcal{F},\mathcal{T})$ and $\sigma \Vdash^{(\mathcal{F},\mathcal{T}_2)} \exists r.\tau$, then $\exists r.\tau$ is a subconcept occurring in $(\mathcal{F},\mathcal{T})$.*

Finally, we show the minimality condition.

Lemma 5. *Let $(\mathcal{F},\mathcal{T}_2)$ be a conservative extension of $(\mathcal{F},\mathcal{T}_{lcs})$ by normalized definitions modulo $(\mathcal{F},\mathcal{T}_{lcs})$. Let D and C be two concepts from $N_{def}^{\mathcal{T}} \cup ExRest$, and let Φ be a concept defined in $\mathcal{T}_2 \setminus \mathcal{T}$. If $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \Phi$ and $C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \Phi$, then $(D,C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \Phi$, for every n.*

Proof. Assume

$$\Phi \equiv \theta_1 \sqcap ... \sqcap \theta_k \sqcap X_1 \sqcap ... \sqcap X_u \sqcap \exists r_1.\Phi_1 \sqcap ... \sqcap \exists r_l.\Phi_l$$

is a definition in $\mathcal{T}_2 \setminus \mathcal{T}$. Here, θ_i, for $i = 1, .., k$, is an element of $Prims$ (from the definition of $(\mathcal{F}, \mathcal{T}_{lcs})$), (and in the case Φ is defined in $\mathcal{T}_2 \setminus \mathcal{T}_{lcs}$, it is a primitive concept). X_i, for $i = 1, .., u$, is a concept defined in $(\mathcal{F}, \mathcal{T}_{lcs})$ (in $\mathcal{T}$ in the case Φ is defined in $\mathcal{T}_{lcs}$), while Φ_i, for $i = 1, ..., l$ is a concept defined in $\mathcal{T}_2 \setminus \mathcal{T}$ (in $\mathcal{T}_{lcs} \setminus \mathcal{T}$ in the case Φ is defined in $\mathcal{T}_{lcs}$). Again, proof is carried out by induction on n. For $n = 0$, $(D, C) \sqsubseteq_0^{(\mathcal{F},\mathcal{T}_2)} \Phi$ follows from the rule (Start).

Assume now that $(D, C) \sqsubseteq_k^{(\mathcal{F},\mathcal{T}_2)} \Phi$ for all $k \leq n$. We prove that $(D, C) \sqsubseteq_{n+1}^{(\mathcal{F},\mathcal{T}_2)} \Phi$. One of the properties of the $\sqsubseteq_n^{(\mathcal{F},\mathcal{T}_{lcs})}$ relation, shown in Lemma 1 in our case yields $(D, C) \sqsubseteq_{n+1}^{(\mathcal{F},\mathcal{T}_2)} \Phi$ iff $(D, C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \theta_1 \sqcap ... \sqcap \theta_k \sqcap X_1 \sqcap ... \sqcap X_u \sqcap \exists r_1.\Phi_1 \sqcap ... \sqcap \exists r_l.\Phi_l$. Therefore, it suffices to prove $(D, C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \theta_1 \sqcap ... \sqcap \theta_k \sqcap X_1 \sqcap ... \sqcap X_u \sqcap \exists r_1.\Phi_1 \sqcap ... \sqcap \exists r_l.\Phi_l$. Again due to (AndR), one way to show this is to give a proof of $(D, C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \theta_i$ for $i = 1, ..., k$, $(D, C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} X_i$ for $i = 1, ..., u$, and $(D, C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\Phi_j$ for $j = 1, ..., l$.

- $(D, C) \sqsubseteq_n \theta_i$: by soundness and completeness of HC, $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \Phi$ implies $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \theta_i$, similarly, $C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \theta_i$, and therefore θ_i occurs on the right-hand side of the definition of (D, C) by Definition 7, since θ_i belongs to $Prims$. Therefore, $(D, C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \theta_i$ follows from completeness of the HC calculus and the fact that $(D, C) \sqsubseteq \theta_i$ holds in all models of $(\mathcal{F}, \mathcal{T})$.
- $(D, C) \sqsubseteq_n X_i$: we distinguish two cases
  1. X_i is defined in $\mathcal{T}$: by soundness and completeness of HC, $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \Phi$ implies $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} X_i$, similarly, $C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} X_i$, and therefore X_i occurs on the right-hand side of the definition of (D, C) by Definition 7. Thus, $(D, C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} X_i$ follows from completeness of the HC calculus and the fact that $(D, C) \sqsubseteq X_i$ holds in all models of $(\mathcal{F}, \mathcal{T})$.
  2. X_i is defined in $\mathcal{T}_{lcs} \setminus \mathcal{T}$: then, X_i is of the form (γ, δ). By soundness and completeness of HC, $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \Phi$ implies $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} (\gamma, \delta)$, similarly, $C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} (\gamma, \delta)$. Now, the induction hypothesis can be applied, since (γ, δ) is defined in $\mathcal{T}_{lcs} \setminus \mathcal{T} \subseteq \mathcal{T}_2 \setminus \mathcal{T}$, and it yields $(D, C) \sqsubseteq_n (\gamma, \delta)$.
- $(D, C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\Phi_j$: again, by soundness and completeness of HC, $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \Phi$ implies $D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\Phi_j$ and $C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \Phi$ implies $C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\Phi_j$. By Lemma 1, this means that there exist concepts α, β and ρ and such that

$$D \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \alpha, \quad \beta \Vdash^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\rho, \quad \rho \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \Phi_j$$
$$C \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \alpha_1, \beta_1 \Vdash^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\rho_1, \rho_1 \sqsubseteq_\infty^{(\mathcal{F},\mathcal{T}_2)} \Phi_j$$

  where $\alpha \sqsubseteq \beta \in \mathcal{F}$ or $D = \alpha = \beta$; $\alpha_1 \sqsubseteq \beta_1 \in \mathcal{F}$ or $C = \alpha_1 = \beta_1$; and ρ and ρ_1 are some concepts occurring in $(\mathcal{F}, \mathcal{T}_2)$.

This further implies $D \sqsubseteq_{\infty}^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\rho$ by applying rule (Concept) to $D \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \alpha$ and $\beta \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\rho$ for every n. Analogously, $C \sqsubseteq_{\infty}^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\rho_1$.

Lemma 4 applied to $\beta \Vdash^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\rho$ and $\beta_1 \Vdash^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\rho_1$ yields the fact that $\exists r_j.\rho$ and $\exists r_j.\rho_1$ are concepts from $(\mathcal{F},\mathcal{T})$. Even more, they are from $ExRest$.

Now, the induction hypothesis can be applied to $\rho \sqsubseteq_{\infty}^{(\mathcal{F},\mathcal{T}_2)} \Phi_j$ and $\rho_1 \sqsubseteq_{\infty}^{(\mathcal{F},\mathcal{T}_2)} \Phi_j$ to obtain $(\rho,\rho_1) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \Phi_j$. On the other hand, by Definition 7, $\exists r_j.(\rho,\rho_1)$ is one of the conjuncts in the definition of (D,C). Now, $(D,C) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \exists r_j.\Phi_j$ can be derived from $(\rho,\rho_1) \sqsubseteq_n^{(\mathcal{F},\mathcal{T}_2)} \Phi_j$, by applying (Ex) rule, (AndL1) or (AndL2) rules several times and (DefL) in the end.

Again, due to the soundness of derivations in HC, considering defined concepts A, B and the corresponding (A,B), we have that (A,B) is subsumed by every concept defined in $\mathcal{T}_2 \setminus \mathcal{T}$ that subsumes both A and B. (We use notation from the previous lemma.) By the comment after Definition 6, this conclusion is sufficient to show property 2 from the definition of hybrid lcs.

Notice also, that, as shown before, the assumption made on the added definitions within the conservative extensions, namely the assumption of them being normalized modulo the TBox, does not cause loss of generality.

Combined with the previously shown property 1 from the definition of hybrid lcs, this proves the following theorem.

Theorem 2. *The concept description (A,B) from the extended hybrid TBox $(\mathcal{F},\mathcal{T}_{lcs})$ is a least common subsumer of A and B w.r.t. the hybrid TBox $(\mathcal{F},\mathcal{T})$.*

5 Conclusion

In this paper, we have described a Gentzen-style calculus for subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes, which is an extension to the case of hybrid TBoxes of the calculi for general TBoxes and for cyclic TBoxes with gfp-semantics that have been introduced in [9]. Based on this calculus, we have developed a polynomial-time decision procedure for subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes. The second result described in this paper was the proof-theoretic computation of least common subsumers w.r.t. hybrid $\mathcal{EL}$-TBoxes. We provide a technique that avoids the undesirable features of normalization. Since the main motivation for considering hybrid TBoxes was that, w.r.t. them, the lcs and msc always exist, the natural next step is to develop a proof-theoretic approach to computing the msc, and we currently investigate that possibility. Other future work in this direction is to try to extend the described techniques to more expressive DLs from the $\mathcal{EL}$ family.

Acknowledgements. The author would like to thank prof. Franz Baader for introducing me into the topic, his valuable advices and supervision of the work, in general.

References

1. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene ontology: tool for the unification of biology. Nat Genet. 25(1), 25–29 (2000)
2. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Proc. IJCAI 2003. Morgan Kaufmann, San Francisco (2003)
3. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proc. IJCAI 2003. Morgan Kaufmann, San Francisco (2003)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. IJCAI 2005. Morgan Kaufmann, San Francisco (2005)
5. Baader, F., Küsters, R., Molitor, R.: Computing least common subsumers in description logics with existential restrictions. In: Proc. IJCAI 1999. Morgan Kaufmann, San Francisco (2003)
6. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: Proc. ECAI 2004. IOS Press, Amsterdam (2004)
7. Brandt, S.: Standard and Non-standard reasoning in Description Logics. Ph.D. dissertation, Institute for Theoretical Computer Science, TU Dresden, Germany (2006)
8. Brandt, S., Model, J.: Subsumption in $\mathcal{EL}$ w.r.t. hybrid TBoxes. In: Furbach, U. (ed.) KI 2005. LNCS (LNAI), vol. 3698. Springer, Heidelberg (2005)
9. Hofmann, M.: Proof-theoretic approach to description-logic. In: Proc. LICS 2005. IEEE Press, Los Alamitos (2005)
10. Model, J.: Subsumtion in $\mathcal{EL}$ bezüglich hybrider TBoxen. Diploma thesis, Institute for Theoretical Computer Science, TU Dresden, Germany (2005)
11. Novakovic, N.: Proof-theoretic approach to subsumption and least common subsumer in $\mathcal{EL}$ w.r.t. hybrid TBoxes. Master thesis, Institute for Theoretical Computer Science, TU Dresden, Germany (2007)
12. Rector, A., Horrocks, I.: Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In: Proc. AAAI 1997. AAAI Press, Menlo Park (1997)
13. Baader, F.: Computing the least common subsumer in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. In: Ganter, B., de Moor, A., Lex, W. (eds.) ICCS 2003. LNCS, vol. 2746, pp. 117–130. Springer, Heidelberg (2003)
14. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation: Experience with SNOMED-RT. Journal of the American Medical Informatics Association (2000); Fall Symposium Special Issue
15. Baader, F., Novakovic, N., Suntisrivaraporn, B.: A Proof-Theoretic Subsumption Reasoner for Hybrid $\mathcal{EL}$-TBoxes. In: Proceedings of the 2008 International Workshop on Description Logics (DL 2008). CEUR-WS (2008)

Extending CARIN to the Description Logics of the $\mathcal{SH}$ Family

Magdalena Ortiz

Institute of Information Systems, Vienna University of Technology
ortiz@kr.tuwien.ac.at

Abstract. This work studies the extension of the *existential entailment algorithm* of CARIN to DLs of the $\mathcal{SH}$ family. The CARIN family of knowledge representation languages was one of the first hybrid languages combining DATALOG rules and Description Logics. For reasoning in one of its prominent variants, which combines $\mathcal{ALCNR}$ with non-recursive DATALOG, the blocking conditions of the standard tableaux procedure for $\mathcal{ALCNR}$ were modified. Here we discuss a similar adaptation to the $\mathcal{SHOIQ}$ tableaux, which provides some new decidability results and tight data complexity bounds for reasoning in non-recursive CARIN, as well as for query answering over Description Logic knowledge bases.

1 Introduction

Description Logics (DLs) are specifically designed for representing structured knowledge in terms of concepts (i.e., classes of objects) and roles (i.e., binary relationships between classes). In the last years, they have evolved into a standard formalism for ontologies which describe a domain of interest in different applications areas. In the context of the Semantic Web, DL-based ontologies have been designated via the Web Ontology Language (OWL) as a standard for describing the semantics of complex Web resources, in order to facilitate access by automated agents. Driven by the need to overcome limitations of DLs and to integrate them into applications, recent research focuses on combining DLs with other declarative knowledge representation formalisms, and in particular with rule-based languages, which play a dominant role in Databases (as query languages) and in Artificial Intelligence [3,8,15,19].

One of the first such *hybrid languages*, CARIN [15], integrates DATALOG programs with some DLs of the $\mathcal{ALC}$ family, being $\mathcal{ALCNR}$ (the basic DL $\mathcal{ALC}$ with number restrictions and role intersection) the most expressive. The limited decidability of hybrid languages was recognised already with the introduction of CARIN, as even very weak DLs yield an undecidable formalism when combined with recursive DATALOG. Three alternatives were proposed to regain decidability: (*i*) the DL constructors causing undecidability are disallowed; (*ii*) only non-recursive rules are allowed; or (*iii*) the variable occurrences in the DL atoms appearing in rules are restricted according to some *safety conditions* that limit their ability to relate unnamed individuals.

In this work, we enhance CARIN with a more expressive DL component and focus on its non-recursive variant (safe rules are briefly discussed in Section 4). We consider the popular DLs of the $\mathcal{SH}$ family, which extend $\mathcal{ALC}$ with role transitivity and

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 324–337, 2008.

containment. The most expressive DL here considered, $\mathcal{SHOIQ}$ (which essentially corresponds to OWL-DL), also supports concepts denoting a single individual called *nominals* ($\mathcal{O}$), inverse roles ($\mathcal{I}$), and qualified number restrictions ($\mathcal{Q}$). By disallowing one of these three constructs, we obtain the expressive and mutually incomparable sublogics known as $\mathcal{SHIQ}$ (corresponding to OWL-Lite), $\mathcal{SHOQ}$, and $\mathcal{SHOI}$ respectively.

For reasoning in non-recursive CARIN, the authors of [15] identified the *existential entailment problem* as a key task and proposed an algorithm for it, based on a tableau (there named *constraint system*) algorithm for satisfiability of $\mathcal{ALCNR}$ knowledge bases with modified blocking conditions. In this way, they also obtained the first algorithm for answering Conjunctive Queries (CQs) and Union of Conjunctive Queries (UCQs) in DLs and for deciding their containment, problems that have become a central topic of interest in recent years. Another central contribution of CARIN was to show a tight coNP upper bound for the aforementioned tasks under *data complexity*, i.e., w.r.t. to the size of the data, assuming that the query/rule component and the terminological part of the knowledge base are fixed. This setting is of major importance, as data repositories can be very large and are usually much larger than the terminology expressing constraints on the data.

In [17] the tableaux algorithm for deciding $\mathcal{SHOIQ}$ knowledge base satisfiability of [11] was adapted following the ideas introduced in [15], to provide an algorithm for the entailment and containment of positive queries in the $\mathcal{SH}$ family of DLs. In this paper we show how this algorithm, analogous to CARIN's existential entailment one, can be exploited for reasoning in non-recursive CARIN and in other hybrid languages. Like [17], the results have two limitations: transitive roles are not allowed in the rule component, and the interaction between number restrictions, inverses and nominals in $\mathcal{SHOIQ}$ may lead to non-termination. However, reasoning is sound and complete if the DL component of the hybrid knowledge base is written in $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ or $\mathcal{SHOI}$, and sound if it is in $\mathcal{SHOIQ}$. We obtain a precise characterisation of the data complexity of reasoning whenever the DATALOG component is non-recursive, and for some cases where it is recursive, e.g., if it satisfies the weak safety conditions of $\mathcal{DL}$+log.

2 Preliminaries

In this section, we define CARIN knowledge bases. The languages that are used in the two components are defined first: DL knowledge bases and DATALOG programs.

Throughout the paper, we consider a fixed alphabet containing the following pairwise disjoint countably infinite sets: a set $\mathbf{C}$ of *DL predicates of arity* 1, called *concept names*; a set $\mathbf{R}$ of *DL predicates of arity* 2, called *role names*, with a subset $\mathbf{R}_+ \subseteq \mathbf{R}$ of *transitive role names*; an alphabet $\mathbf{P}$ of *rule predicates*, where each $p \in \mathbf{P}$ has an associated arity $m \geq 0$; a set $\mathbf{I}$ of *individuals*; and a set $\mathbf{V}$ of *variables*. This alphabet is used for defining knowledge bases, whose semantics is given by (first-order) interpretations.

Definition 1 (Interpretation). *An* interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *is given by a non-empty* domain $\Delta^{\mathcal{I}}$ *and an* interpretation function $\cdot^{\mathcal{I}}$ *that maps each predicate* $p \in \mathbf{P} \cup \mathbf{C} \cup \mathbf{R}$ *of arity* n *to a subset of* $(\Delta^{\mathcal{I}})^n$*, and each individual in* $\mathbf{I}$ *to an element of* $\Delta^{\mathcal{I}}$.

2.1 Description Logics

The DL $\mathcal{SHOIQ}$ and its sublogics $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ are defined as usual.[1]

Definition 2 ($\mathcal{SHOIQ}$ Knowledge Bases). *A* role expression R *(or simply* role*) is a role name $P \in \mathbf{R}$ or its* inverse P^-. *A* role inclusion axiom *is an expression $R \sqsubseteq R'$, where R and R' are roles. A* role hierarchy $\mathcal{R}$ *is a set of role inclusion axioms.*

As usual, $\mathsf{Inv}(R) = P^-$ *if* $R = P$ *for some* $P \in \mathbf{R}$ *and* $\mathsf{Inv}(R) = P$ *if* $R = P^-$. *For a role hierarchy $\mathcal{R}$, the relation $\sqsubseteq^*_{\mathcal{R}}$ denotes the reflexive, transitive closure of $\sqsubseteq$ over $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(R') \mid R \sqsubseteq R' \in \mathcal{R}\}$. We write* $\mathsf{Trans}(R, \mathcal{R})$ *if* $R \sqsubseteq^*_{\mathcal{R}} R'$ *and* $R' \sqsubseteq^*_{\mathcal{R}} R$ *for some* $R' \in \mathbf{R}_+ \cup \{R^- \mid R \in \mathbf{R}_+\}$. *A role S is* simple *w.r.t. $\mathcal{R}$ if for no role R with* $\mathsf{Trans}(R, \mathcal{R})$ *we have that* $R \sqsubseteq^*_{\mathcal{R}} S$.

Let $a, b \in \mathbf{I}$ be individuals, $A \in \mathbf{C}$ a concept name, C and C' concepts, $P \in \mathbf{R}$ a role name, R a role, S a simple role, and $n \geq 0$ an integer. Concepts are defined inductively according to the following syntax:

$$C, C' \longrightarrow A \mid \{a\} \mid C \sqcap C' \mid C \sqcup C' \mid \neg C \mid \forall R.C \mid \exists R.C \mid \geq n\, S.C \mid \leq n\, S.C$$

Concepts of the form $\{a\}$ are called nominals. *A* concept inclusion axiom *is an expression $C \sqsubseteq D$. An* assertion *is an expression $A(a)$, $P(a, b)$ or $a \not\approx b$. A* TBox *is a finite set of concept inclusion axioms, and an* ABox *is a finite set of assertions. A ($\mathcal{SHOIQ}$)* knowledge base *(KB) is a triple $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a TBox, $\mathcal{R}$ is a role hierarchy, and $\mathcal{A}$ is an ABox.*[2]

Definition 3 ($\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$ Knowledge Bases). *Roles and concepts in $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$ are defined as in $\mathcal{SHOIQ}$, except that*

- *in $\mathcal{SHOQ}$, the inverse role constructor P^- is not available;*
- *in $\mathcal{SHIQ}$, nominals $\{a\}$ are not available;*
- *in $\mathcal{SHOI}$, number restrictions $\geq n\, S.C$, $\leq n\, S.C$ are not available,*

For $\mathcal{L}$ one of $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, or $\mathcal{SHOI}$, an $\mathcal{L}$ knowledge base *is a $\mathcal{SHOIQ}$ knowledge base $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ such that all roles and concepts occurring in it are in $\mathcal{L}$.*

Definition 4 (Semantics of DL KBs). *Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation such that $R^{\mathcal{I}} = (R^{\mathcal{I}})^+$ for each $R \in \mathbf{R}_+$. To interpret K, the interpretation function is inductively extended to complex concepts and roles as follows:*

$$\begin{array}{ll}
(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & (\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} & (\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\} \\
(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} & (\leq n\, R.C)^{\mathcal{I}} = \{x \mid |\{y \mid (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \leq n\} \\
(P^-)^{\mathcal{I}} = \{(y, x) \mid (x, y) \in P^{\mathcal{I}}\} & (\geq n\, R.C)^{\mathcal{I}} = \{x \mid |\{y \mid (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \geq n\}
\end{array}$$

$\mathcal{I}$ satisfies *an assertion α, denoted $\mathcal{I} \models \alpha$, if $\alpha = A(a)$ implies $a^{\mathcal{I}} \in A^{\mathcal{I}}$, $\alpha = P(a, b)$ implies $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in P^{\mathcal{I}}$ and $\alpha = a \not\approx b$ implies $a^{\mathcal{I}} \neq b^{\mathcal{I}}$; $\mathcal{I}$ satisfies a role inclusion*

[1] For the sake of uniformity, we use the name $\mathcal{SHOI}$ instead of the also common $\mathcal{SHIO}$.

[2] Note that only concepts and role names may occur in $\mathcal{A}$, but this is no limitation. Indeed, for a complex C, an assertion $C(a)$ can be expressed by $A_c(a)$ and an axiom $A_c \sqsubseteq C$ in $\mathcal{T}$, while an assertion $R^-(x, y)$ can be replaced by $\mathsf{Inv}(R)(a, b)$.

axiom $R \sqsubseteq R'$ *if* $R^{\mathcal{I}} \subseteq R'^{\mathcal{I}}$*, and a concept inclusion axiom* $C \sqsubseteq C'$*, if* $C^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$*.* $\mathcal{I}$ satisfies a role hierarchy $\mathcal{R}$ *and* a terminology $\mathcal{T}$*, if it satisfies every axiom of* $\mathcal{R}$ *and* $\mathcal{T}$ *respectively. Furthermore,* $\mathcal{I}$ satisfies an ABox $\mathcal{A}$*, if it satisfies every assertion in* $\mathcal{A}$*. Finally,* $\mathcal{I}$ *is a* model *of* $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$*, denoted* $\mathcal{I} \models K$*, if it satisfies* $\mathcal{T}$*,* $\mathcal{R}$*, and* $\mathcal{A}$*.*

2.2 DATALOG

We now define DATALOG programs and their semantics, also given by interpretations.[3]

Definition 5 (DATALOG rules and DATALOG programs). *A* (rule/DL) atom *is an expression* $p(\overline{x})$*, where* p *is a (rule/DL) predicate, and* $\overline{x}$ *is a tuple from* $\mathbf{V} \cup \mathbf{I}$ *of the same arity as* p*. If* $\overline{x} \subseteq \mathbf{I}$*, then* $p(\overline{x})$ *is* ground*.*

A DATALOG rule *is an expression of the form* $q(\overline{x}) :- p_1(\overline{y_1}), \ldots, p_n(\overline{y_n})$ *where* $n \geq 0$*,* $q(\overline{x})$ *is a rule atom, each* $p_i(\overline{y_i})$ *is an atom, and* $\overline{x} \cap \mathbf{V} \subseteq \overline{y_1} \cup \ldots \cup \overline{y_n}$*. As usual,* $q(\overline{x})$ *is called the* head *of the rule, and* $p_1(\overline{y_1}), \ldots, p_n(\overline{y_n})$ *is called the* body*. A rule with* $n = 0$ *is called a* fact *and can be written simply* $q(\overline{x})$*.*

A DATALOG program $\mathcal{P}$ *is a set of* DATALOG *rules. Its* dependency graph *is the directed graph whose nodes are the predicates* p *occurring in* $\mathcal{P}$ *with an edge* $p \rightarrow p'$ *if* p' *occurs in the head and* p *in the body of a rule in* $\mathcal{P}$*.* $\mathcal{P}$ *is* recursive *if its dependency graph contains some cycle, and* non-recursive *otherwise.*

Definition 6 (Semantics of DATALOG Programs). *An interpretation* $\mathcal{I}$ satisfies a ground atom $p(\overline{a})$*, written* $\mathcal{I} \models p(\overline{a})$*, if* $(\overline{a})^{\mathcal{I}} \in p^{\mathcal{I}}$*. A* substitution *is a mapping* $\sigma : \mathbf{V} \cup \mathbf{I} \rightarrow \Delta^{\mathcal{I}}$ *with* $\sigma(a) = a^{\mathcal{I}}$ *for every* $a \in \mathbf{I}$*. For an atom* $p(\overline{x})$ *and a substitution* σ*, we say that* σ makes $p(\overline{x})$ true in $\mathcal{I}$*, in symbols* $\mathcal{I}, \sigma \models p(\overline{x})$*, if* $\mathcal{I} \models p(\sigma(\overline{x}))$*. We say that* $\mathcal{I}$ satisfies a rule r*, denoted* $\mathcal{I} \models r$*, if every substitution that makes all the atoms in the body true also makes the atom in the head true. If* $\mathcal{I} \models r$ *for each* $r \in \mathcal{P}$*, then* $\mathcal{I}$ is a model of $\mathcal{P}$*, in symbols* $\mathcal{I} \models \mathcal{P}$*.*

2.3 CARIN Knowledge Bases

Now we define the CARIN language. In what follows, $\mathcal{L}$ denotes a DL of the $\mathcal{SH}$ family.

Definition 7 (CARIN knowledge bases). *A* CARIN-$\mathcal{L}$ knowledge base *is a tuple* $\langle K, \mathcal{P} \rangle$ *where* K *is an* $\mathcal{L}$ *knowledge base, called the* DL component *of* K*, and* $\mathcal{P}$ *is a* DATALOG *program, called its* rule (or DATALOG) component*. A* CARIN-$\mathcal{L}$ *knowledge base is (non-) recursive if its rule component* $\mathcal{P}$ *is (non-) recursive.*

Note that only rule predicates can occur in the head of rules of $\mathcal{P}$. This is a common feature of many hybrid languages that assume that the DL knowledge base provides a commonly shared conceptualisation of a domain, while the rule component defines application-specific relations that can not change the structure of this conceptual model.

The semantics of CARIN KBs arises naturally from the semantics of its components. As in the original CARIN, we define as main reasoning task the entailment of a ground atom, which may be either a DL assertion or a DATALOG ground fact.

[3] Note that we consider first-order semantics, without the minimality requirement.

Definition 8 (CARIN-$\mathcal{L}$ entailment problem). *An interpretation $\mathcal{I}$ is a* model *of a* CARIN-$\mathcal{L}$ *knowledge base $\mathcal{K} = \langle K, \mathcal{P}\rangle$, in symbols $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I} \models K$ and $\mathcal{I} \models \mathcal{P}$. For a ground atom α, $\mathcal{K} \models \alpha$ denotes that $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models \alpha$ for every $\mathcal{I}$. The* CARIN-$\mathcal{L}$ entailment problem *is to decide, given $\mathcal{K}$ and α, whether $\mathcal{K} \models \alpha$.*

We note that the standard DL reasoning tasks (e.g., KB consistency and subsumption) are reducible to entailment in CARIN, as the latter generalises instance checking.

3 Reasoning in Non-recursive CARIN

In this section, we provide an algorithm for reasoning in non-recursive CARIN. The key to the decidability in this variant of CARIN is the limited interaction between the DL and rule predicates. Indeed, if we have a non-recursive DATALOG program $\mathcal{P}$ and we want to verify entailment of an atom $p(\overline{a})$, it is sufficient to consider the rules in $\mathcal{P}$ whose head predicate is p and unfold them into a set of rules where only $p(\overline{a})$ occurs in the head, and the bodies contain only DL atoms and ground facts. The CARIN-$\mathcal{L}$ entailment problem with such a restricted rule component is then reducible to the *entailment of UCQs.*

The *query entailment* (or informally, query answering) problem is DLs has gained much attention in recent times. Many papers have studied the problem of answering CQs and UCQs over DL knowledge bases, e.g., [1,5,6,14,20]. We consider the more expressive language of positive existential queries.

3.1 Non-recursive CARIN and Query Entailment

We introduce *positive (existential) queries* (PQs), which generalise CQs and UCQs.[4]

Definition 9 (Positive Queries, Query Entailment). *A* positive (existential) query *(PQ) over a KB K is a formula $\exists\overline{x}.\varphi(\overline{x})$, where $\overline{x}$ is a vector of variables from $\mathbf{V}$ and $\varphi(\overline{x})$ is built using $\wedge$ and $\vee$ from DL atoms whose variables are in $\overline{x}$. If $\varphi(\overline{x})$ is a conjunction of atoms then $\exists\overline{x}.\varphi(\overline{x})$ is a* conjunctive query *(CQ); if $\varphi(\overline{x})$ is in disjunctive normal form then it is a* union of conjunctive queries *(UCQ).*

Let $Q = \exists\overline{x}.\varphi(\overline{x})$ be a PQ over K and let $\mathcal{I}$ be an interpretation. For a substitution σ, let Q^σ be the Boolean expression obtained from φ by replacing each atom α with $\top$ if $\mathcal{I}, \sigma \models \alpha$, and with $\bot$ otherwise. We call σ a match for $\mathcal{I}$ and Q, *denoted $\mathcal{I}, \sigma \models Q$, if Q^σ evaluates to $\top$. $\mathcal{I}$* is a model of Q, *written $\mathcal{I} \models Q$, if $\mathcal{I}, \sigma \models Q$ for some σ.*

We say that K entails *Q, denoted $K \models Q$, if $\mathcal{I} \models Q$ for each model $\mathcal{I}$ of K. The* query entailment problem *is to decide, given K and Q, whether $K \models Q$.*

Note that a PQ can be rewritten into an equivalent, possibly exponentially larger, UCQ.

The UCQ (and thus PQ) entailment problem and CARIN entailment problem are closely related. In fact, we can reduce the former to the latter as follows:

Proposition 1. *Let K be a $\mathcal{SHOIQ}$ knowledge base and let $Q = \exists\overline{x}.\varphi_1(\overline{x_1}) \vee \ldots \vee \varphi_n(\overline{x_n})$ be a UCQ over K. Then $\mathcal{K} \models Q$ iff $\langle K, \mathcal{P}\rangle \models q$, where $q \in \mathbf{P}$ is fresh, $\mathcal{P}$ is the* DATALOG *program containing the rules $q :- \varphi_i'(\overline{x_i})$ for each $1 \leq i \leq n$, and each $\varphi_i'(\overline{x_i})$ is obtained from $\varphi_i(\overline{x_i})$ by replacing each connective $\wedge$ by a comma.*

[4] We consider Boolean queries, to which non-Boolean ones can be reduced as usual, and disregard the difference between the equivalent query entailment and query answering problems.

We show next that the converse also holds, i.e., the CARIN-$\mathcal{SHOIQ}$ entailment problem can be reduced to query entailment over the DL component. As a consequence, whenever we have a procedure for deciding query entailment, we obtain a sound and complete algorithm for reasoning in non-recursive CARIN.

Definition 10 (Rule unfolding and program depth). *Given two* DATALOG *rules:*

$$r_1 = q_1(\overline{x_1}) :- p_1(\overline{y_1}), \ldots, p_n(\overline{y_n}), \quad \textit{and} \quad r_2 = q_2(\overline{x_2}) :- p'_1(\overline{y'_1}), \ldots, p'_m(\overline{y'_m}),$$

where $q_2 = p_i$ for some $1 \leq i \leq n$, let θ be the most general unifier of $\overline{x_2}$ and $\overline{y_i}$. Then the following rule r' is an unfolding of r_2 in r_1*:*

$$q_1(\theta\overline{x_1}) :- p_1(\theta\overline{y_1}), \ldots, p_{i-1}(\theta\overline{y_{i-1}}), p'_1(\theta\overline{y'_1}), \ldots, p'_m(\theta\overline{y'_m}), p_{i+1}(\theta\overline{y_{i+1}}), \ldots p_n(\theta\overline{y_n}).$$

The width *of a rule r, denoted* $\mathsf{width}(r)$*, is the number of atoms in its body. The* depth *of a non-recursive* DATALOG *program $\mathcal{P}$, written* $\mathsf{depth}(\mathcal{P})$*, is $w + 1$, where w is the width of the longest rule that can be obtained from some rule in $\mathcal{P}$ by repeatedly unfolding in it other rules of $\mathcal{P}$, until no more unfoldings can be applied. If $\mathcal{P} = \emptyset$,* $\mathsf{width}(r) = 1$.

Note that $\mathsf{depth}(\mathcal{P})$ is finite and can be effectively computed, as $\mathcal{P}$ is non-recursive.

Definition 11 (Unfolding). *The* unfolding *of a non-recursive* DATALOG *program $\mathcal{P}$ for a ground rule atom $p(\overline{a})$ is the program $\mathcal{P}_{p(\overline{a})}$ obtained as follows:*

(1) Let $\mathcal{P}_1$ denote the set of rules in $\mathcal{P}$ where the head is of the form $p(\overline{x})$ and there is a unifier of of $\overline{a}$ and $\overline{x}$. $\mathcal{P}_2$ is the set of rules $p(\theta\overline{x}) :- q_1(\theta\overline{y_1}), \ldots, q_n(\theta\overline{y_n})$ where $p(\overline{x}) :- q_1(\overline{y_1}), \ldots, q_n(\overline{y_n}) \in \mathcal{P}_1$ and θ is the most general unifier of $\overline{a}$ and $\overline{x}$.
(2) For a rule r, let $r_\mathcal{P}$ denote the set of unfoldings in r of a rule from $\mathcal{P}$ (note that it may be empty). Apply exhaustively the following rule: if $r \in \mathcal{P}_2$ and the body of r contains a rule atom α such that $\alpha \notin \mathcal{P}$, replace r by $r_\mathcal{P}$ in $\mathcal{P}_2$. The resulting program is $\mathcal{P}_{p(\overline{a})}$.

Every model of $\mathcal{P}$ is also a model of $\mathcal{P}_{p(\overline{a})}$. Intuitively, $\mathcal{P}_{p(\overline{a})}$ captures the part of $\mathcal{P}$ that is relevant for the entailment of $p(\overline{a})$. Each rule in $\mathcal{P}_{p(\overline{a})}$ has $p(\overline{a})$ as head, and its body contains only DL atoms and ground facts from $\mathcal{P}$, which are true in every model of $\mathcal{P}$. Due to this restricted form, $\mathcal{P}_{p(\overline{a})}$ can easily be transformed into an equivalent UCQ.

Definition 12 (Query for a ground atom). *The* query for a ground atom α w.r.t. a non-recursive DATALOG program $\mathcal{P}$*, denoted $U_{\mathcal{P},\alpha}$, is the UCQ defined as follows:*

- If α is a DL atom, then $U_{\mathcal{P},\alpha} = \alpha$.
- Otherwise $U_{\mathcal{P},\alpha} = \exists\overline{x}.Q_1 \vee \ldots \vee Q_m$, where $r_1 \ldots r_m$ are the rules of $\mathcal{P}_\alpha$, each Q_i is the conjunction of the DL atoms in the body of r_i, and $\overline{x}$ contains the variables of each Q_i.

Note that if a rule atom α occurs as a fact in $\mathcal{P}$, it also occurs as a fact in $\mathcal{P}_\alpha$, and $U_{\mathcal{P},\alpha}$ is trivially true (since it has an empty disjunct which is always true). If $\mathcal{P}_\alpha = \emptyset$ then $U_{\mathcal{P},\alpha}$ is always false; this is the case, e.g., if α does not unify with the head of any rule.

Proposition 2. *Let $\mathcal{K} = \langle K, \mathcal{P} \rangle$ be a non-recursive* CARIN-$\mathcal{SHOIQ}$ *knowledge base and let α be a ground atom. Then $\mathcal{K} \models \alpha$ iff $K \models U_{\mathcal{P},\alpha}$.*

3.2 A Tableaux Algorithm for Query Entailment

We have shown that the non-recursive CARIN-$\mathcal{SHOIQ}$ entailment problem can be reduced to the entailment of a PQ (in fact, a UCQ suffices). In this section, we describe the algorithm given in [17] to solve the latter for the $\mathcal{SH}$ family DLs. Provided that the query contains only simple roles, it is sound and complete for $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$; for $\mathcal{SHOIQ}$ it is sound, but termination remains open.

The algorithm is an extension of the one in [15] for the *existential entailment problem*, which informally speaking, simultaneously captures UCQ entailment and CQ/UCQ containment (i.e., given a CQ Q_1 and a UCQ Q_2, decide whether $K \models Q_1$ implies $K \models Q_2$). We present it as a query entailment algorithm: this suffices for reasoning in non-recursive CARIN and the generalisation to containment is trivial. A first extension to CQs in $\mathcal{SHIQ}$ was presented in [18]. Here we recall the extension to PQs in $\mathcal{SHOIQ}$ of [17], where the reader may find detailed definitions, proofs and examples.

We build on [11] and use *completion graphs*, finite relational structures that represent models of a $\mathcal{SHIQ}$ knowledge base K. After an initial completion graph $\mathcal{G}_K$ for K is built, new completion graphs are generated by repeatedly applying *expansion rules*. Every model of K is represented in some completion graph that results from the expansion, thus $K \models Q$ can be decided by considering a suitable set of such graphs.

In what follows, $K{=}\langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$ denotes a $\mathcal{SHOIQ}$ knowledge base; the set of roles occurring in K and their inverses is denoted $\mathbf{R}_K$. A denotes a concept name; D, E denote concepts; R, R' denote roles; and a, b denote individuals.

A *completion graph* $\mathcal{G}$ for K comprises a finite labelled directed graph whose nodes $\mathsf{nodes}(\mathcal{G})$ are labelled by concepts and whose arcs $\mathsf{arcs}(\mathcal{G})$ are labelled by roles. The nodes in $\mathsf{nodes}(\mathcal{G})$ are of two kinds: *individual nodes* and *variable nodes*. The label of each individual node contains some nominal $\{a\}$ indicating that the node stands for the individual $a \in \mathbf{I}$. A variable node contains no nominal concepts and represents one or more unnamed individuals whose existence is implied by the knowledge base. An additional binary relation is used to store explicit inequalities between the nodes of $\mathcal{G}$.

In a completion graph $\mathcal{G}$, each arc $v \to w$ is labelled with a set $\mathcal{L}(v \to w)$ of roles from $\mathbf{R}_K$ and each node v is labelled with a set $\mathcal{L}(v)$ of 'relevant' concepts. The set of all the relevant concepts is denoted by $\mathsf{clos}(K)$ and contains the standard concept closure of $\neg C \sqcup D$ for each axiom $C \sqsubseteq D$ in the knowledge base K (closed under subconcepts and their negations) and some additional concepts that may be introduced by the rules (e.g., to correctly ensure the propagation of the universal restrictions, concepts of the form $\forall R'.D$ for some $\forall R.D \in \mathsf{clos}(K)$ and R' a transitive subroles of R are used, so they are also included in the closure).

The usual relations between the nodes in a completion graph $\mathcal{G}$ are defined as in [11,17]: if $v \to w \in \mathsf{arcs}(\mathcal{G})$, then w is a *successor* of v and v a *predecessor* of w. The transitive closures of successor and predecessor are *ancestor* and *descendant* respectively. If $R' \in \mathcal{L}(v \to w)$ for some role R' with $R' \sqsubseteq^* R$, then w is an R-*successor* of v. We call w an R-*neighbour* of v, if w is an R-*successor* of v, or if v is an $\mathsf{Inv}(R)$-*successor* of w. The *distance* between two nodes in $\mathcal{G}$ is defined in the natural way.

The *initial completion graph* $\mathcal{G}_K$ for K contains a node a labelled $\mathcal{L}(a) = \{\{a\}\} \cup \{\neg C \sqcup D \,|\, C \sqsubseteq D \in \mathcal{T}\} \cup \{\neg C \sqcup D \,|\, C \sqsubseteq D \in \mathcal{T}_\mathcal{A}\}$ for each individual a in K, where

$\mathcal{T}_{\mathcal{A}} = \{\{a\} \sqsubseteq A \mid A(a) \in \mathcal{A}\} \cup \{\{a\} \sqsubseteq \exists P.\{b\} \mid P(a,b) \in \mathcal{A}\} \cup \{\{a\} \sqsubseteq \neg\{b\} \mid a \neq b \in \mathcal{A}\}$ is a set of concept inclusion axioms representing the assertions in $\mathcal{A}$.

We apply *expansion rules* to the initial $\mathcal{G}_K$ and obtain new completion graphs. The rules may introduce new variable nodes, but they are always successors of exactly one existing node. Hence the variable nodes form a set of trees that have individual nodes as roots. Some of these variable nodes may have an individual node as a successor, thus a tree can have a path ending with an arc to an individual node.

Blocking conditions are given to ensure that the expansion stops after sufficiently many steps. They are inspired by [15], but adapted to these more expressive logics, and depend on a depth parameter $n \geq 0$, generalising the non-parametrised blocking of [11]. This blocking is the crucial difference between our algorithm and [11]. According to the blocking conditions of [11], the expansion of a completion graph $\mathcal{G}$ terminates when a node v with a predecessor u is reached such that there is some ancestor u' of u that has in turn a successor v' such that the pairs (u', v') and (u, v) have the same node-arc-node labels, i.e., when a pair of nodes that is isomorphic to a previously existing one appears in $\mathcal{G}$. This *pairwise blocking* condition ensures that the expansion stops when $\mathcal{G}$ already represents a model of $\mathcal{K}$. If the knowledge base is satisfiable, then there is a way to non-deterministically apply the expansion rules until this blocking occurs, and a completion graph that represents a model of the knowledge base is obtained.

Since we want to decide query entailment, this is not enough: we need to obtain a set of models that suffices to check query entailment. Our modified blocking ensures that a completion graph is blocked only if it represents a set of models that are indistinguishable by the query. Instead of halting the expansion when a previously occurred *pair* of nodes appears, we stop when a repeated instance of an n-graph occurs, where the n-graph of a node v is a tree of variable nodes of depth at most n rooted at v, plus arcs to the individual nodes that are direct successors of a node in this tree. We now define formally this modified blocking. The next definition is technically quite involved. It is taken from [17], where more explanations and some examples can be found.

Definition 13 (n-graph blocking). *Given an integer $n \geq 0$ and a completion graph $\mathcal{G}$, let $\mathsf{vn}(\mathcal{G})$ denote the set of variable nodes in $\mathcal{G}$. The* blockable n-graph of node $v \in \mathsf{vn}(\mathcal{G})$ *is the subgraph $\mathcal{G}^{n,v}$ of $\mathcal{G}$ that contains v and (i) every descendant $w \in \mathsf{vn}(\mathcal{G})$ of v within distance n, and (ii) every successor $w' \in \mathsf{in}(\mathcal{G})$ of each such w. If w has in $\mathcal{G}^{n,v}$ no successors from $\mathsf{vn}(\mathcal{G})$, we call w a* leaf of $\mathcal{G}^{n,v}$. *Nodes v, v' of $\mathcal{G}$ are* n-graph equivalent via a bijection ψ from $\mathsf{nodes}(\mathcal{G}^{n,v})$ to $\mathsf{nodes}(\mathcal{G}^{n,v'})$ *if (1) $\psi(v) = v'$; (2) for every $w \in \mathsf{nodes}(\mathcal{G}^{n,v})$, $\mathcal{L}(w) = \mathcal{L}(\psi(w))$; (3) $\mathsf{arcs}(\mathcal{G}^{n,v'}) = \{\psi(w) \to \psi(w') \mid w \to w' \in \mathsf{arcs}(\mathcal{G}^{n,v})\}$; and (4) for every $w \to w' \in \mathsf{arcs}(\mathcal{G}^{n,v})$ $\mathcal{L}(w \to w') = \mathcal{L}(\psi(w) \to \psi(w'))$.*

Let $v, v' \in \mathsf{vn}(\mathcal{G})$ be n-graph equivalent via ψ, where both v and v' have predecessors in $\mathsf{vn}(\mathcal{G})$, v' is an ancestor of v in $\mathcal{G}$, and v is not in $\mathcal{G}^{n,v'}$. If v' reaches v on a path containing only nodes in $\mathsf{vn}(\mathcal{G})$, then v' is a n-witness of v in $\mathcal{G}$ via ψ. *Moreover, $\mathcal{G}^{n,v'}$* graph-blocks *$\mathcal{G}^{n,v}$* via *ψ, and each $w \in \mathsf{nodes}(\mathcal{G}^{n,v'})$* graph-blocks via *$\psi$ the node $\psi^{-1}(w)$ in $\mathcal{G}^{n,v}$.*

Let ψ be a bijection between two subgraphs G', G of $\mathcal{G}$ such that G' graph-blocks G via ψ. A node $v \in \mathsf{nodes}(\mathcal{G})$ is n-blocked*, if $v \in \mathsf{vn}(\mathcal{G})$ and v is either directly or indirectly n-blocked; v is* indirectly n-blocked*, if one of its ancestors is n-blocked; v*

is directly n-blocked *iff none of its ancestors is n-blocked and v is a leaf of G; in this case we say that v is (directly) n-blocked by $\psi(v)$. An R-neighbour w of a node v in $\mathcal{G}$ is* n-safe *if $v \in \mathsf{vn}(\mathcal{G})$ or if w is not n-blocked.*

Note that v is m-blocked for each $m \leq n$ if it is n-blocked. When $n \geq 1$, then n-blocking implies pairwise blocking.

The expansion rules are analogous to the ones in [11], where 'blocked' is replaced by 'n-blocked' and'safe' is replaced by 'n-safe'. Due to space restrictions, we can not present the expansion rules here, but they can be found in [17].

A *clash* in a completion graph $\mathcal{G}$ is an explicit contradiction (e.g.,$\{A, \neg A\} \subseteq \mathcal{L}(v)$ for some node v), and it indicates that $\mathcal{G}$ represents an empty set of models and thus the expansion can stop. If $\mathcal{G}$ does not contain a clash it is called *clash-free*. If $\mathcal{G}$ contains a clash or no more rules are applicable to it, then we say that it is *n-complete*. We denote by $\mathbb{G}_K$ the set of completion graphs that can be obtained from the initial $\mathcal{G}_K$ via the expansion rules, and by $\mathsf{ccf}_n(\mathbb{G}_K)$ the ones that are n-complete and clash free.

We view each graph in $\mathbb{G}_K$ as a representation of a (possibly infinite) set of models of K. Intuitively, the models of K are all the relational structures containing $\mathcal{A}$ that satisfy the constraints given by $\mathcal{T}$ and $\mathcal{R}$. Each completion graph $\mathcal{G}$ contains the initial $\mathcal{A}$ and additional constraints, implicit in $\mathcal{T}$ and $\mathcal{R}$, that were explicated by applying the rules. When there is more than one way to apply a rule to a graph $\mathcal{G}$ (e.g. in the $\sqcup$-rule either C_1 or C_2 can be added), the models represented by $\mathcal{G}$ are 'partitioned' into the sets of models represented by each of the different graphs that can be obtained.[5]

Importantly, every model of K is represented by some $\mathcal{G}$ in $\mathbb{G}_K$. Thus, the union of all the models of the graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$ coincides with all the models of K, independently of the value of n. Therefore, in order to decide query entailment, we can choose an arbitrary $n \geq 0$ and check all the models of all the completion graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$. This is still not enough to yield a decision procedure: although the set $\mathsf{ccf}_n(\mathbb{G}_K)$ is finite, we do not have an algorithm for deciding entailment of query Q in all (possibly infinitely many) models of a completion graph $\mathcal{G}$. However, if a suitable n is chosen, the latter can be effectively decided by finding a syntactic mapping of the query into $\mathcal{G}$.

Definition 14 (Query mapping). *Let $Q = \exists \overline{x}.\varphi(\overline{x})$ be a PQ and let $\mathcal{G}$ be a completion graph. Let $\mu : \mathsf{Vl}(Q) \to \mathsf{nodes}(\mathcal{G})$ be a total function such that $\{a\} \in \mathcal{L}(\mu(a))$ for each individual a in $\mathsf{Vl}(Q)$. We write $C(x) \hookrightarrow_\mu \mathcal{G}$ if $C \in \mathcal{L}(\mu(x))$, and $S(x, x') \hookrightarrow_\mu \mathcal{G}$ if $\mu(x')$ is an S-neighbour of $\mu(x)$. Let γ be the Boolean expression obtained from $\varphi(\overline{x})$ by replacing each atom α in φ with $\top$, if $\alpha \hookrightarrow_\mu \mathcal{G}$, and with $\bot$ otherwise. We say that μ is a* mapping for Q into $\mathcal{G}$, *denoted $Q \hookrightarrow_\mu \mathcal{G}$, if γ evaluates to $\top$. Q* can be mapped into *$\mathcal{G}$, denoted $Q \hookrightarrow \mathcal{G}$, if there is a mapping μ for Q into $\mathcal{G}$.*

It is not hard to see that if $Q \hookrightarrow \mathcal{G}$, then there is a mapping for Q in every model represented by $\mathcal{G}$. The converse is slightly more tricky and only holds if Q contains only simple roles and if a suitable value for the blocking parameter n is chosen. Very roughly, n-blocking ensures that all paths of length $\leq n$ that occur in the models of K are already found in some $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$. Since the query contains only simple roles, matches for

[5] This view slightly differs from the more common one (e.g., [11]) in which a completion graph is a representation of one single model (the one obtained from $\mathcal{G}$ by standard unravelling).

Q in a model do not require paths larger than the number $nr(Q)$ of role atoms in the largest disjunct when Q is transformed into a UCQ (which is in turn bounded by the total role atoms in Q). As a consequence, Q can not distinguish models that are equivalent up to $nr(Q)$-blocking. The following theorem is shown in [17]:

Theorem 1. *Let Q be a positive query where only simple roles occur, let K be a $\mathcal{SHOIQ}$ KB, and let $n \geq nr(Q)$. Then $K \models Q$ iff $Q \hookrightarrow \mathcal{G}$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$.*

The theorem suggests to verify PQ entailment as follows: (*i*) obtain all the completion graphs in $\mathsf{ccf}_{nr(Q)}(\mathbb{G}_K)$, and (*ii*) check each of them for query mappability. This yields a decision procedure provided that both steps can be effectively executed. We show below that this is the case if the KB is in any of $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$.

Due to Proposition 2, we can use the same decision procedure for the CARIN-$\mathcal{L}$ entailment problem. For any atom α, the number of atoms in each disjunct in $U_{\mathcal{P},\alpha}$ is bounded by $\mathsf{depth}(\mathcal{P})$. Trivially, if only simple roles occur in $\mathcal{P}$, the same holds for $U_{\mathcal{P},\alpha}$. Therefore, from Proposition 2 and Theorem 1, we easily obtain:

Corollary 1. *Let α be a ground atom, let $\mathcal{K} = \langle K, \mathcal{P}\rangle$ be a non-recursive* CARIN-$\mathcal{SHOIQ}$ *knowledge base where only simple roles occur in $\mathcal{P}$, and let $n \geq \mathsf{depth}(\mathcal{P})$. Then $\mathcal{K} \models \alpha$ iff $U_{\mathcal{P},\alpha} \hookrightarrow \mathcal{G}$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$.*

Note that the outlined decision procedure requires that, for each given input the query α, $U_{\mathcal{P},\alpha}$ is built by unfolding $\mathcal{P}$. If several atoms are to be evaluated, a more efficient alternative can be to obtain the completion graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$ and then evaluate all the rules of the program over each graph, in a bottom-up way. Roughly, for a completion graph $\mathcal{G}$ and a program $\mathcal{P}$, we can obtain the smallest set $S(\mathcal{G}, \mathcal{P})$ of atoms that contains all the DL ground facts entailed by $\mathcal{G}$, and that contains the head of a rule r whenever there is a match of the body atoms to the atoms in the set $S(\mathcal{G}, \mathcal{P})$ (under suitable substitutions). It is not hard to see that, for every atom α, $\alpha \in S(\mathcal{G}, \mathcal{P})$ iff $K \models U_{\alpha,\mathcal{P}}$.[6]

4 Complexity of Reasoning in Hybrid KBs

We have shown that we can effectively solve the non-recursive-CARIN and the PQ entailment problem whenever we have an effective procedure or obtaining the graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$ and deciding query mappability for each of them. The latter is trivially decidable if each $\mathcal{G}$ and the set $\mathsf{ccf}_n(\mathbb{G}_K)$ are finite (e.g., by traversing, for each $\mathcal{G}$, the finitely many possible mappings from the query variables to the nodes of $\mathcal{G}$).

As for the first part, it was shown in [17] that the expansion of an initial $\mathcal{G}_K$ into the set $\mathsf{ccf}_n(\mathbb{G}_K)$ terminates if there is no interaction between the number restrictions, inverses and nominals. Roughly, whenever variable nodes can cause a number restriction to be violated at an individual node a, the so-called $o?$-rule is applied to generate new individual neighbours for a. This rule is never applicable for $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$ KBs, allowing us to prove termination. For $\mathcal{SHOIQ}$, however, due to the mutual dependency between the depth of the forest and the number of individual nodes generated by the $o?$-rule that results from our modified blocking, we cannot ensure that

[6] This procedure has the same worst-case complexity as the one outlined above.

it terminates (although we believe that, using the prioritised strategy for rule application of [11], it will do so in many cases).

The following bounds for the modified tableaux algorithm were shown in [17], while in the CARIN-entailment setting they are analysed in more detail in [16]. Given a KB $K{=}\langle\mathcal{T},\mathcal{R},\mathcal{A}\rangle$ and PQ Q, $||K,Q||$ denotes the combined size of the strings encoding the K and Q (assuming unary encoding of numbers in the number restrictions), and $|\mathcal{A}|$ the number of assertions in $\mathcal{A}$. Similarly, $||\mathcal{P}||$ denotes the size the string encoding a given DATALOG program $\mathcal{P}$.

Proposition 3. *The expansion of $\mathcal{G}_K$ into some $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$, $n \geq 0$, terminates in time triple exponential in $||K,Q||$ if n is polynomial in $||K,Q||$. If n is a constant and Q and all of K except $\mathcal{A}$ are fixed, then it terminates in time polynomial in $|\mathcal{A}|$.*

The same bounds apply to the number of nodes in each $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$. Checking whether $Q \hookrightarrow \mathcal{G}$ can be easily done in time single exponential in the size of Q and polynomial in $|\mathsf{nodes}(\mathcal{G})|$; if Q is fixed, $Q \hookrightarrow \mathcal{G}$ can be tested in time polynomial in the size of $\mathcal{G}$ (as there are only polynomially many candidate assignments).

Theorem 2. *The PQ entailment problem is decidable if the input KB is in any of $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ and only simple roles occur in the query. Furthermore, it can be refuted non-deterministically in time polynomial in the size of the ABox.*

A matching lower bound holds already for instance checking in the very weak $\mathcal{AL}$ [1].

Theorem 3. *For every DL extending $\mathcal{AL}$ and contained in $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, or $\mathcal{SHOI}$, deciding the entailment of a PQ in which only simple roles occur has* CONP*-complete data complexity.*

For a non-recursive DATALOG program $\mathcal{P}$, $\mathsf{depth}(\mathcal{P})$ is finite and effectively computable, and $nr(U_{\alpha,\mathcal{P}}) \leq \mathsf{depth}(\mathcal{P})$. Although $\mathsf{depth}(\mathcal{P})$ is single exponential in $||\mathcal{P}||$, it is constant if $\mathcal{P}$ is fixed. Hence we obtain:

Theorem 4. *The non-recursive* CARIN-$\mathcal{L}$ *entailment problem is decidable if $\mathcal{L}$ is any of $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ and only simple roles occur in the rule component of the KB. Furthermore, it has* CONP*-complete data complexity if $\mathcal{L}$ is a DL extending $\mathcal{AL}$.*

This result provides an exact characterisation of the data complexity of the non-recursive CARIN-$\mathcal{L}$ entailment problem for a wide range of description logics. Unfortunately, our work does not provide optimal upper bounds with respect to the combined complexity. In fact, the tableaux algorithm from [11] on which our work is based terminates in non-deterministic double exponential time in the worst case, even if the input is a $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ or $\mathcal{SHOI}$ knowledge base whose satisfiability problem is known to be EXPTIME-complete [21,7,9]. This suboptimality carries on to our results. Additionally, our reduction from the CARIN-entailment problem to UCQ entailment causes an exponential blow-up whose inevitability has not been explored.

The $\mathcal{DL}$+log Family. In [19], Rosati introduced the $\mathcal{DL}$+*log* family of formalisms coupling arbitrary DLs with DATALOG rules. It allows for recursive programs and, in

order to preserve decidability, imposes some *weak safety* conditions on the rules which are a relaxed version of CARIN's safety.

An $\mathcal{L}$+log knowledge base is composed of a knowledge base in the DL $\mathcal{L}$ and a set of weak-safe DATALOG rules (possibly with disjunction and negation as failure). Its decidability depends on the one of query containment in $\mathcal{L}$: as shown in [19] (Theorem 11), satisfiability in $\mathcal{L}$+log is decidable iff CQ/UCQ containment is decidable in $\mathcal{L}$.[7] From well known results that relate query containment and query answering, it follows that our method can be exploited for deciding this problem.

Theorem 5. *Satisfiability of an $\mathcal{L}$+log knowledge base is decidable if $\mathcal{L}$ is $\mathcal{SHOQ}$, $\mathcal{SHIQ}$ or $\mathcal{SHOI}$ and the* DATALOG *component contains only simple roles.*

Furthermore, it follows from [19] that whenever the data complexity of query entailment is strictly lower than that of reasoning in the rule component, the latter carries on to the overall data complexity of reasoning. As a consequence, it can also be concluded from our results that reasoning in the above setting has Σ_2^p-complete data complexity when the DATALOG component is a disjunctive program with negation.

Related Complexity Results. Since this work started, many query answering algorithms have been proposed and new complexity bounds have been found. Due to Proposition 2 (which is independent of the particular DL in the DL component), the new decidability results for answering UCQs in DLs imply new decidability boundaries for non-recursive CARIN, similarly as the decidability of CQ/UCQ containment carries on to the $\mathcal{DL}$+log setting. Furthermore, the data complexity of UCQ answering can be directly transferred to non-recursive CARIN.

From this and recent results, the decidability statements in Theorems 4 and 5 holds also in the presence of transitive roles in the query if $\mathcal{L}$ is $\mathcal{SHOQ}$ [6], $\mathcal{SHIQ}$ [5], or $\mathcal{ALCQIb}_{reg}$, another expressive DL [2]. Further interesting results can be obtained from the latter, which is to our knowledge the most general algorithm for query answering in DLs without nominals. In particular, the DL known as $\mathcal{SRIQ}$ [10] (closely related to the DL $\mathcal{SROIQ}$ underlying OWL 2) can be reduced to (a minor extension of) $\mathcal{ALCQIb}_{reg}$. Exploiting the regular expressions in the query atoms in [2], one can use that algorithm to decide PQ entailment and containment in $\mathcal{SRIQ}$ (note that containment of queries with regular expressions does not follow from [2] in general, but it does if the query on the left is a plain CQ); hence both CARIN and $\mathcal{DL}$+log are decidable for $\mathcal{SRIQ}$. We also note that the algorithm in [2] provides an optimal 2EXPTIME upper bound for satisfiability of $\mathcal{SRIQ}$ knowledge bases; this was reported open in [12].

As for data complexity, CONP completeness for CARIN entailment, $\mathcal{DL}$+log satisfiability and UCQ answering (with arbitrary query/rule component) for $\mathcal{SHIQ}$ follows from [5]. Most recently, the PTIME-complete data complexity of PQ answering in Horn-$\mathcal{SHIQ}$ (a disjunction-free fragment of $\mathcal{SHIQ}$) was established [4]; this carries on to both non-recursive CARIN entailment and $\mathcal{DL}$+log satisfiability. To our knowledge, no other tight bounds for the $\mathcal{SH}$ family have been established. CARIN entailment and $\mathcal{DL}$+log satisfiability (with a positive DATALOG component) are PTIME-complete for $\mathcal{EL}$ and some of its extensions contained in $\mathcal{EL}^{++}$ [14] and $\mathcal{ELI}^f$ [13], see also [20].

[7] In general, 'satisfiability' means under both FOL and NM semantics.

Finally, due to the results in [1], the non-recursive CARIN entailment problem is in LOGSPACE for the DLs of the DL-Lite family; their PTIME-completeness had already been established for $\mathcal{DL}$+log [19].

5 Conclusion

In this paper, we have presented an algorithm for the CARIN entailment problem in knowledge bases that combine a $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ or $\mathcal{SHOI}$ KB with a non-recursive DATALOG program containing only simple roles. It relies on a tableaux-based algorithm for positive query entailment which builds on the techniques from [11] and generalises the existential entailment algorithm given in [15] for a DL which is far less expressive than $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$.

For the three mentioned sublogics of $\mathcal{SHOIQ}$, our algorithm is worst-case optimal in data complexity, and allows us to characterise the data complexity of reasoning with non-recursive DATALOG programs for a wide range of DLs, including very expressive ones. Namely, for all DLs of the $\mathcal{SH}$ family except $\mathcal{SHOIQ}$, the problem has CONP-complete data complexity, and is thus not harder than instance checking in $\mathcal{AL}$.

Combining the aforementioned DLs with recursive DATALOG results in an undecidable formalism. However, our results can be combined with those of Rosati [19] to show decidability if the rules are weakly safe and the query contains no transitive roles. Further decidability and data complexity results for reasoning in hybrid languages can be obtained from the reduction of non-recursive CARIN to UCQ entailment presented here and from the results in [19], some of them were discussed in this work.

Acknowledgements. This paper presents some results obtained during the author's Masters thesis and complements those in [18,17]. The author wants to express her great gratitude to her thesis supervisors, Thomas Eiter and Diego Calvanese. She also thanks the JELIA organisers for the invitation to present these results, as well as the consortium of the European Masters in Computational Logic. This work was partially supported by the Austrian Science Fund (FWF) grant P20840, and the Mexican National Council for Science and Technology (CONACYT) grant 187697.

References

1. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of KR 2006, pp. 260–270 (2006)
2. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics: An automata-theoretic approach. In: Proc. of AAAI 2007, pp. 391–396 (2007)
3. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with Description Logics for the Semantic Web. In: Proc. KR 2004, pp. 141–151 (2004)
4. Eiter, T., Gottlob, G., Ortiz, M., Simkus, M.: Query Answering in the Description Logic Horn-$\mathcal{SHIQ}$. In: Proc. of JELIA (to appear, 2008)
5. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In: Proc. of IJCAI 2007, pp. 399–404 (2007)
6. Glimm, B., Horrocks, I., Sattler, U.: Conjunctive query entailment for $\mathcal{SHOQ}$. In: Proc. of DL 2007, vol. 250, pp. 65–75 (2007)

7. Glimm, B., Horrocks, I., Sattler, U.: Deciding $\mathcal{SHOQ}$ plus role conjunction knowledge base consistency using alternating automata. In: Proc. of DL 2008 (2008)
8. Heymans, S., Vermeir, D.: Integrating ontology languages and answer set programming. In: Proceedings DEXA Workshops 2003, pp. 584–588. IEEE Computer Society, Los Alamitos (2003)
9. Hladik, J.: A tableau system for the description logic $\mathcal{SHIO}$. In: Proceedings of IJCAR Doctoral Programme. CEUR Workshop Proceedings, vol. 106, CEUR-WS.org (2004)
10. Horrocks, I., Kutz, O., Sattler, U.: The irresistible $\mathcal{SRIQ}$ (2005)
11. Horrocks, I., Sattler, U.: A tableaux decision procedure for $\mathcal{SHOIQ}$. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp. 448–453 (2005)
12. Kazakov, Y.: $\mathcal{SRIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Proc. DL 2008 (2008)
13. Krisnadhi, A., Lutz, C.: Data complexity in the $\mathcal{EL}$ family of description logics. In: Proc. of DL 2007 (2007)
14. Krötzsch, M., Rudolph, S., Hitzler, P.: Conjunctive queries for a tractable fragment of OWL 1.1. In: Proc. of ISWC/ASWC 2007, pp. 310–323 (2007)
15. Levy, A.Y., Rousset, M.-C.: Combining Horn rules and description logics in CARIN. Artificial Intelligence 104(1–2), 165–209 (1998)
16. Ortiz, M., Calvanese, D., Eiter, T.: Data Complexity of Query answering in Expressive Description Logics via Tableaux. INFSYS Research Report 1843-07-07. Vienna University of Technology (November 2007)
17. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics via tableaux. In: J. of Automated Reasoning (June 2008)
18. Ortiz, M., Calvanese, D., Eiter, T.: Characterizing data complexity for conjunctive query answering in expressive description logics. In: Proc. of AAAI 2006 (2006)
19. Rosati, R.: DL+log: Tight integration of description logics and disjunctive datalog. In: Proc. of KR 2006, pp. 68–98 (2006)
20. Rosati, R.: On conjunctive query answering in $\mathcal{EL}$. In: Proc. of DL 2007 (2007)
21. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen (2001)

How to Restore Compactness into Probabilistic Logics?

Aleksandar Perović[1], Zoran Ognjanović[2],
Miodrag Rašković[2], and Zoran Marković[2]

[1] Saobraćajni fakultet, Vojvode Stepe 305, 11000 Beograd, Srbija
pera@sf.bg.ac.yu
[2] Matematički institut SANU, Kneza Mihaila 36, 11000 Beograd, Sbija
{zorano,miodragr,zoranm}@mi.sanu.ac.yu

Abstract. The paper offers a proof of the compactness theorem for the ${}^*\mathbb{R}$-valued polynomial weight formulas. We also provide a complete axiomatization for the polynomial weight formulas.

1 Introduction

One of the main proof-theoretical issues when someone analyzes a logic concerns providing an axiom system and proving its completeness. There are two kinds of completeness theorems:

- The simple completeness theorem ('every consistent formula is satisfiable') and
- The extended completeness theorem ('every consistent set of formulas is satisfiable').

In the presence of the compactness theorem (every finitely satisfiable set of formulas is satisfiable), extended completeness is a consequence of the simple completeness.

Probabilistic logics formalize uncertain reasoning. The standard approach [2,4,7,8,9,10,11,12,13,14,16,18] adds probability operators to classical logic. The paper [2] was the first one containing some recursive axiom systems for probabilistic logics and the corresponding simple completeness proofs. For so called linear weight formulas a simple complete axiomatization was given, together with a companion NP complete decision procedure. The case of polynomial weight formulas was also discussed, but without any axiomatization. However, some papers [11,18] that followed [2] pointed out the non-compactness phenomena of the proposed real-valued probabilistic semantics. It implies a big logical problem: there are consistent sets that are not satisfiable, where 'consistent set' means that contradiction is not derivable from the set using the assumed axiom system. For example, using the notation that will be introduced below (where $w(p)$ can be read as 'probability of the primitive proposition p'), every finite subset of the set $T = \{\neg w(p) = 0\} \cup \{w(p) < 1/n : n \text{ is a positive integer}\}$ is satisfiable, while

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 338–348, 2008.

the set itself is not. However, T is consistent with respect to the axiom systems from [2].

There are several possible ways to overcome this problem. First, one can consider probabilistic functions with a fixed finite range [11,14,18]. In that case the set T is not even finitely satisfiable. From the syntactical point of view, the following axiom:

$$\bigvee_{r \in I} w(\alpha) = r,$$

where $I = \{0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}, 1\}$ is a fixed set, guarantees the finiteness of the range of probabilistic functions. The next possibility is to allow non-recursive axiomatization with infinitary inference rules [10,11,16]. An example is:

From the set of premises

$$\{\varphi \rightarrow w(\alpha) \geqslant r - \frac{1}{k} \mid k \in \mathbb{N} \wedge k \geqslant \frac{1}{r}\}$$

infer $\varphi \rightarrow w(\alpha) \geqslant r$.

The rule intuitively says that if the probability is arbitrarily close to r, then it is at least r. It has two purposes: to eliminate the possibility of non-archimedean probability functions, and to allow construction of the canonical model. Note that, in this approach, the above set T becomes inconsistent.

The aim of this paper is to analyze another possibility: how to restore compactness into probabilistic logics by using non-archimedean probabilistic functions. In other words, we will consider ${}^*\mathbb{R}$-valued probabilities. We will show that in that case compactness holds and discuss the corresponding consequences. We would like to note that ${}^*\mathbb{R}$-valued probabilities were also used in [6] to model default reasoning.

In [16], the logic with nonstandard (hyperreal) probabilistic semantics was discussed, in order to model default reasoning by means of conditional probabilities. The strong completeness of the introduced formalism was achieved by means of another kind of infinitary inference rule:

From the set of premises

$$\{\varphi \rightarrow P_{\neq s}\alpha \mid s \in [0,1]_{\mathbb{F}}\}$$

infer $\neg\phi$.

Here $[0,1]_{\mathbb{F}}$ is a unit interval in some countable non-Archimedean field $\mathbb{F}$ (for instance, $\mathbb{F}$ can be a Hardy field $\mathbb{Q}(\varepsilon)$, where ε is a proper infinitesimal), and $P_{\neq s}\alpha$ reads "the probability of α is not equal to s". Notice that the above rule provides that the range of any measure appearing in arbitrary measurable model (for definition see Section 3) is a subset of $[0,1]_{\mathbb{F}}$.

It is interesting to point out that certain fuzzy logics, such as $\text{Ł}\Pi\frac{1}{2}$, can be combined with probabilistic modalities in order to obtain a logic for reasoning about conditional probabilities. Namely, both Łukasiewicz and product implications behave like standard ordering of the reals (more precisely, product implication behaves like a truncated division), product conjunction behaves like the usual

multiplication of the reals, Łukasiewicz disjunction behaves like a truncated sum, and all rational numbers from the real unit interval $[0,1]$ are definable in Ł$\Pi\frac{1}{2}$. In particular, all polynomial weight inequalities with coefficients from $[0,1]\cap\mathbb{Q}$ can be formally expressed in such logic, which makes it slightly less expressive than the logic of polynomial weight formulas (semantically introduced in [2]). A finitary strong completeness theorem of mixed probability–Ł$\Pi\frac{1}{2}$ logic was proved in [7], while the strong completeness theorem with respect to non-Archimedean measures was proved in [3].

The rest of the paper is organized as follows. In section 2 and 3 we introduce syntax and semantics of logics with linear and polynomial weight formulas. In the section 4 we prove the compactness theorem for the ${}^*\mathbb{R}$-valued polynomial weight formulas. Theorem 5, also proved in the section 4, is somewhat stronger form of the compactness theorem for the ${}^*\mathbb{R}$-valued linear weight formulas. In the section 5 we discuss the consequences of the previously proved compactness, and provide a simply-complete axiomatization for the polynomial weight formulas. Concluding remarks are in the section 6.

2 Syntax

Let $\mathcal{P}=\{p_n \mid n<\omega\}$ be the set of propositional letters. By $For_{\mathcal{P}}$ we will denote the set of all Boolean combinations of propositional letters, and we will refer to it as the set of propositional formulas. The variables for propositional letters are α, β and γ, indexed if necessary.

The set LWT of linear weight terms is recursively defined as follows:

1. $LWT_0=\{0,1\}\cup\{w(\alpha)\mid \alpha\in For_{\mathcal{P}}\}$
2. $LWT_{n+1}=LWT_n\cup\{(f+g)\mid f,g\in LWT_n\}\cup\{(-f)\mid f\in LWT_n\}$
3. $LWT=\bigcup\limits_{n<\omega} LWT_n$.

To simplify notation, we will use the usual abbreviations. For instance,

$$w(\alpha)-2w(\beta)+3$$

is the abbreviated version of the linear weight term

$$((w(\alpha)+((-w(\beta))+(-w(\beta))))+((1+1)+1)).$$

Linear weight terms will be denoted by f, g and h, indexed if necessary.

The set PWT of polynomial weight terms is recursively defined as follows:

1. $PWT_0=\{0,1\}\cup\{w(\alpha)\mid \alpha\in For_{\mathcal{P}}\}$
2. $PWT_{n+1}=PWT_n\cup\{(\mathtt{f}+\mathtt{g}),(\mathtt{f}\cdot\mathtt{g}),(-\mathtt{f})\mid \mathtt{f},\mathtt{g}\in PWT_n\}$
3. $PWT=\bigcup\limits_{n<\omega} PWT_n$.

Clearly, $LWT\subseteq PWT$. Again, we will use the usual abbreviations. For example,

$$2w(\alpha)^2w(\beta)-3$$

is the abbreviated version of

$$((((w(\alpha) \cdot w(\alpha)) \cdot w(\beta)) + ((w(\alpha) \cdot w(\alpha)) \cdot w(\beta))) + (((-1) + (-1)) + (-1))).$$

The variables for polynomial weight terms are $\mathtt{f}$, $\mathtt{g}$ and $\mathtt{h}$, indexed if necessary.

Definition 1. *A basic linear weight formula is any formula of the form*

$$f \geqslant 0,$$

where $f \in LWT$. The set LWF of linear weight formulas is the set of all Boolean combinations of the basic linear weight formulas. □

Definition 2. *A basic polynomial weight formula is any formula of the form*

$$\mathtt{f} \geqslant 0,$$

where $\mathtt{f} \in PWT$. The set PWF of polynomial weight formulas is the set of all Boolean combinations of the basic polynomial weight formulas. □

Polynomial weight formulas will be denoted by ϕ, ψ and θ, indexed if necessary. As above, we will use the usual abbreviations in order to simplify notation. Some examples are listed below.

- $\mathtt{f} \geqslant \frac{1}{3}$ is $3\mathtt{f} - 1 \geqslant 0$.
- $\mathtt{f} \leqslant 0$ is $-\mathtt{f} \geqslant 0$.
- $\mathtt{f} > 0$ is $\neg(\mathtt{f} \leqslant 0)$.
- $\mathtt{f} < 0$ is $\neg(\mathtt{f} \geqslant 0)$.
- $\mathtt{f} = 0$ is $\mathtt{f} \leqslant 0 \ \wedge \ \mathtt{f} \geqslant 0$.
- $\mathtt{f} \neq 0$ is $\neg(\mathtt{f} = 0)$.
- $\mathtt{f} \geqslant \mathtt{g}$ is $\mathtt{f} - \mathtt{g} \geqslant 0$. Similarly are defined $\mathtt{f} \leqslant \mathtt{g}$, $\mathtt{f} > \mathtt{g}$, $\mathtt{f} < \mathtt{g}$, $\mathtt{f} = \mathtt{g}$ and $\mathtt{f} \neq \mathtt{g}$.

3 Semantics

By ${}^*\mathbb{R}$ we will denote some ω_1-saturated elementary extension of the ordered field of the reals. Let ${}^*[0,1]$ be the unit interval in ${}^*\mathbb{R}$ (see [5,17]). A *model* is any quadruple $M = \langle S, H, \mu, v\rangle$ with the following properties:

1. S is a nonempty set.
2. $H \subseteq P(W)$ is an algebra of sets on S.
3. $\mu : For_{\mathcal{P}} \longrightarrow {}^*[0,1]$ is a finitely additive probability measure.
4. $v : For_{\mathcal{P}} \times W \longrightarrow \{0,1\}$ is a truth evaluation.

If M is any model and $\alpha \in For_{\mathcal{P}}$, then let

$$[\alpha]_M = \{s \in S \mid v(\alpha, s) = 1\}.$$

We will omit M from the subscript whenever the context is clear. A model M is *measurable* if $[\alpha] \in H$ for all α.

Definition 3. *Let M be any measurable model and let $\phi \in PWF$. We define the satisfiability relation $M \models \phi$ recursively on the complexity of ϕ as follows:*

1. $M \models \mathtt{f} \geqslant 0$ *if* $\mathtt{f}^M \geqslant 0$, *where:*
   (a) $0^M = 0$, $1^M = 1$
   (b) $w(\alpha)^M = \mu([\alpha])$
   (c) $(\mathtt{f} + \mathtt{g})^M = \mathtt{f}^M + \mathtt{g}^M$
   (d) $(\mathtt{f} \cdot \mathtt{g})^M = \mathtt{f}^M \cdot \mathtt{g}^M$.
   (e) $(-\mathtt{f})^M = -\mathtt{f}^M$.
2. $M \models \psi \wedge \theta$ *if* $M \models \psi$ *and* $M \models \theta$.
3. $M \models \neg\psi$ *if* $M \not\models \psi$. □

The notions of satisfiability and validity are introduced in the usual way.

4 Compactness

In this section we will prove the compactness theorem for the polynomial weight formulas, and also prove that each finitely satisfiable theory in LWF has a model whose measure is a hyper-time valued. We remind the reader that a hyper-time interval is any set of the form $\{0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{K-1}{K}, 1\}$, where K is an infinite natural number.

Theorem 4. *Suppose that $T \subseteq PWF$ is finitely-satisfiable. Then, T is satisfiable.*

Proof. Let $\mathcal{L}_{OF} = \{+, \cdot, \leqslant, 0, 1\}$ be the language of the ordered fields and let

$$\mathcal{L} = \mathcal{L}_{OF} \cup \{w(\alpha) \mid \alpha \in For_{\mathcal{P}}\},$$

where each $w(\alpha)$ is a new symbol for a constant. Clearly, T may be seen as a set of Σ_0 $\mathcal{L}$-sentences, i.e., each sentence in T is a quantifier-free formula without variables. We define an $\mathcal{L}$-theory Γ as a union of the following $\mathcal{L}$-theories:

1. RCF (the theory of real closed fields)
2. T
3. $\{w(\alpha) = 1 \mid \alpha$ is a tautology$\}$
4. $\{w(\neg\alpha) = 1 - w(\alpha) \mid \alpha \in For_{\mathcal{P}}\}$
5. $\{w(\alpha) = w(\beta) \mid \alpha \leftrightarrow \beta$ is a tautology$\}$
6. $\{w(\alpha \vee \beta) = w(\alpha) + w(\beta) - w(\alpha \wedge \beta) \mid \alpha, \beta \in For_{\mathcal{P}}\}$.

Due to the compactness theorem for the first order logic, to show that Γ is consistent we only need to show that Γ is finitely satisfiable.

Let $\{\phi_1, \ldots, \phi_n\} \subseteq T$. Since T is finitely satisfiable (as a PWF-theory), there is a model M which satisfies $\phi_1, \ldots, \phi_n$. Now we can define $\mathcal{L}$-model $\mathcal{M}$ as follows:

– The universe of $\mathcal{M}$ is ${}^*\mathbb{R}$.
– $\mathcal{L}_{OF}$ is interpreted in $\mathcal{M}$ in the same way as in ${}^*\mathbb{R}$.
– Each $w(\alpha)$ is interpreted in $\mathcal{M}$ as $\mu([\alpha])$.

Clearly, $\mathcal{M} \models (\Gamma \setminus T) \cup \{\phi_1, \ldots, \phi_n\}$, so Γ is finitely satisfiable.

By the downward Löwenheim-Skolem-Tarski theorem, Γ has a countable model $\mathcal{N}$. Since ${}^*\mathbb{R}$ is ω_1-saturated, there is an elementary embedding (with respect to the $\mathcal{L}_{OF}$) $F : \mathcal{N} \longrightarrow {}^*\mathbb{R}$. Finally, we can define the model of T in the following way:

– S is the set of all classical propositional models
– $v(\alpha, s) = 1$ if $s(\alpha) = 1$
– $H = \{[\alpha] \mid \alpha \in For_{\mathcal{P}}\}$
– $\mu([\alpha]) = F(w(\alpha)^{\mathcal{N}})$. □

Theorem 5. *Suppose that $T \subseteq LWF$ is finitely satisfiable. Than, T has a model $M = \langle S, H, \mu, v\rangle$ such that* $\mathrm{rng}(\mu) \subseteq \{0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{K-1}{K}, 1\}$*, where $K = K'!$ and $K' \in {}^*\mathbb{N} \setminus \mathbb{N}$.*

Proof. First of all, we will need some additional notation. Given linear weight formula ϕ, by $\|\phi\|$ we will denote the maximal $m < \omega$ such that p_m appears in ϕ. The set of all finite sequences (functions of the form $a : n \longrightarrow 2$) of 0's and 1's will be denoted by ${}^{<\omega}2$. Here $0 = \emptyset$, $1 = \{0\}$, $2 = \{0, 1\}$ etc. If $|a| = n > 0$, then p^a is the formula

$$p_0^{a_0} \wedge \cdots \wedge p_{n-1}^{a_{n-1}},$$

where p_i^0 is $\neg p_i$ and p_i^1 is p_i. Let $T = \{\phi_n \mid n < \omega\}$. Without any loss of generality, we may assume that $\phi_{n+1} \rightarrow \phi_n$ is a valid formula for all n (instead of T, we can consider the theory $\{\phi_0, \phi_0 \wedge \phi_1, \phi_0 \wedge \phi_1 \wedge \phi_2, \ldots\}$). Furthermore, we may assume that each $w(\alpha)$ that appears in ϕ_n has a form $w(p^a)$, where $a : \|\phi_n\| \longrightarrow 2$.

Now we are ready to proceed with the proof. For each $n < \omega$, let A_n be the set of all $x : {}^{\leqslant K}2 \longrightarrow \{0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{K-1}{K}, 1\}$, where ${}^{\leqslant K}2$ denotes the set of all binary sequences of length at most K, with the following properties:

1. For each $a : \|\phi_n\| \longrightarrow 2$, formula ϕ_n (in the sense of the language $\mathcal{L}$ introduced in the proof of the theorem 4) is satisfied in the valuation $w(p^a) \mapsto x(a)$.
2. $\sum_{a \in {}^{\|\phi_n\|}2} x(a) = 1$.
3. For each $a \in {}^{<\|\phi_n\|}2$,

$$x(a) = \sum_{b \in {}^{\|\phi_n\| - |a|}2} x(a^\frown b),$$

where $a^\frown b$ is the concatenation of the sequences a and b.

Notice that if $x \in A_n$, then each measurable model $\langle S, H, \mu, v\rangle$ such that

$$\mu[p^a] = x(a), \quad a \in {}^{<\|\phi_n\|}2,$$

is a model of $\phi_0 \wedge \cdots \wedge \phi_n$ (by assumption, ϕ_n is equivalent with $\phi_0 \wedge \cdots \wedge \phi_n$).

It is easy to see that each A_n is a nonempty internal set (in the sense of the non-standard analysis). Indeed, the fact that A_n is a internal set immediately follows from the definition of A_n. To see that A_n is nonempty, we will use the finite satisfiability of T: let $\langle S, H, \mu, v\rangle \models \phi_n$. Define $x : {}^{\leqslant \|\phi_n\|}2 \longrightarrow \{0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{K-1}{K}\}$ by

$$x(a) = \mu[p^a], \quad a \in {}^{\leqslant \|\phi_n\|}2.$$

The first item of the definition of A_n is satisfied by x since $\langle S, H, \mu, v\rangle$ is a model of ϕ_n. The remaining two items follow from the fact that μ is a finitely additive probability measure.

Since A_n, $n < \omega$ is a descending sequence of nonempty internal sets, by the ω_1 saturation principle, $\bigcap\limits_{n<\omega} A_n \neq \emptyset$. Let $x \in \bigcap\limits_{n<\omega} A_n$. Then we can define the model $M = \langle S, H, \mu, v\rangle$ of T in the following way:

- S is the set of all classical truth evaluations.
- $v(\alpha, s) = 1$ if $s(\alpha) = 1$.
- $H = \{[\alpha] \mid \alpha \in For_{\mathcal{P}}\}$.
- For each $\alpha \in For_{\mathcal{P}}$ let $I(\alpha)$ be the unique subset of ${}^{\|\alpha\|}2$ such that the formula

  $$\alpha \leftrightarrow \bigvee_{a \in I(\alpha)} p^a$$

  is a tautology. Then, let

  $$\mu([\alpha]) = \sum_{a \in I(\alpha)} x(a).$$

By the choice of x, the above construction is correct. □

5 Axiomatization

Since we proved the compactness theorem for the polynomial weight formulas, in order to provide the strongly complete axiomatization of LWF and PWF logics, we only need to provide a simply complete axiomatization. In the case of linear weight formulas, we can use the well known axiomatization of Fagin, Halpern and Megiddo given in [2].

The case of polynomial weight formulas is much more difficult. On the one hand, the PWF logic is decidable, so we can trivially axiomatize it by the single axiom schemata:

- ϕ, whenever ϕ is a valid formula.

In this trivial case we do not need any inference rule and the simple completeness immediately holds. The key issue here is how to obtain a non-trivial finitary axiomatization which is simply complete. To provide such an axiomatization,

we will need to investigate the structure of valid formulas with respect to their complexity.

The next lemma is an immediate consequence of the definition of the satisfiability relation $\models$.

Lemma 6. *Suppose that T is an arbitrary set of polynomial weight formulas. Then:*

1. *(L$\rightarrow$) $T \models \phi \rightarrow \psi$ iff $T \cup \{\phi\} \models \psi$.*
2. *(L$\neg\neg$) $T \models \neg\neg\phi$ iff $T \models \phi$.*
3. *(L$\neg\rightarrow$) $T \models \neg(\phi \rightarrow \psi)$ iff $T \models \phi$ and $T \models \neg\psi$.*
4. *(R$\neg\neg$) $T \cup \{\neg\neg\phi\} \models \psi$ iff $T \cup \{\phi\} \models \psi$.*
5. *(R$\neg\rightarrow$) $T \cup \{\neg(\phi \rightarrow \psi)\} \models \theta$ iff $T \cup \{\phi, \neg\psi\} \models \theta$.*
6. *(R$\rightarrow$) $T \cup \{\phi \rightarrow \psi\} \models \theta$ iff $T \cup \{\neg\theta\} \models \phi$ and $T \cup \{\neg\theta\} \models \neg\psi$.*

We can use L and R "rules" of the previous lemma to reduce the complexity of sub-formulas of the given probabilistic formula ϕ, and consequently, equivalently reduce $\models \phi$ to the finite conjunction of sequents of the form

$$\{\pm(\mathtt{f}_1 \geqslant 0), \cdots, \pm(\mathtt{f}_n \geqslant 0)\} \models \pm(\mathtt{g} \geqslant 0).$$

To make the reduction strategy clearer, we give the following two examples.

Example 7.

$$\begin{array}{ll} & \models (\mathtt{f} \geqslant 0 \rightarrow \mathtt{g} \geqslant 0) \rightarrow \mathtt{h} \geqslant 0 \\ \text{iff} & \{\mathtt{f} \geqslant 0 \rightarrow \mathtt{g} \geqslant 0\} \models \mathtt{h} \geqslant 0 \qquad (\text{L}\rightarrow) \text{ of Lemma 6} \\ \text{iff} & \{\mathtt{h} < 0\} \models \mathtt{f} \geqslant 0 \text{ and } \{\mathtt{h} < 0\} \models \mathtt{g} < 0 \quad (\text{R}\rightarrow) \text{ of Lemma 6} \end{array}$$

Example 8.

$$\begin{array}{ll} & \models (\neg\psi \rightarrow \neg\phi) \rightarrow (\phi \rightarrow \psi) \\ \text{iff} & \{\neg\psi \rightarrow \neg\phi\} \models \phi \rightarrow \psi \qquad (\text{L}\rightarrow) \text{ of Lemma 6} \\ \text{iff} & \{\neg\psi \rightarrow \neg\phi, \phi\} \models \psi \qquad (\text{L}\rightarrow) \text{ of Lemma 6} \\ \text{iff} & \{\neg\psi, \phi\} \models \neg\psi \text{ and } \{\neg\psi, \phi\} \models \neg\neg\phi \quad (\text{R}\rightarrow) \text{ of Lemma 6} \\ \text{iff} & \{\neg\psi, \phi\} \models \neg\psi \text{ and } \{\neg\psi, \phi\} \models \phi \qquad (\text{L}\neg) \text{ of Lemma 6} \end{array}$$

Now we are ready to define the axiomatization Ax_{PWF} of polynomial weight formulas.

5.1 Propositional Axioms

This group of axioms contains two axiom schemata:

A1. All instances of tautologies.
A2. All valid formulas of the form

$$\pm(\mathtt{f}_1 \geqslant 0) \wedge \cdots \wedge \pm(\mathtt{f}_n \geqslant 0) \rightarrow \pm(\mathtt{g} \geqslant 0),$$

where $\mathtt{f}_1, \ldots, \mathtt{f}_n, \mathtt{g}$ are in the so called *normal form*

$$\sum_i k_i \prod_{a \in {}^m 2} (w(p^a))^{l_{ai}}.$$

5.2 Probabilistic Axioms

This group of axioms contain five axiom schemata:

A3 $0 \leqslant w(\alpha) \leqslant 1$.
A4 $w(\alpha) = 1$, whenever α is a tautology.
A5 $w(\neg\alpha) = 1 - w(\alpha)$.
A6 $w(\alpha) = w(\beta)$, whenever $\alpha \leftrightarrow \beta$ is a tautology.
A7 $w(\alpha \vee \beta) = w(\alpha) + w(\beta) - w(\alpha \wedge \beta)$.

5.3 Algebraic Axioms

This group of axioms contains the following axiom schemata:

A8 $\mathtt{f} + \mathtt{g} = \mathtt{g} + \mathtt{f}$.
A9 $(\mathtt{f} + \mathtt{g}) + \mathtt{h} = \mathtt{f} + (\mathtt{g} + \mathtt{h})$.
A10 $\mathtt{f} + 0 = \mathtt{f}$.
A11 $\mathtt{f} - \mathtt{f} = 0$.
A12 $\mathtt{f} \cdot \mathtt{g} = \mathtt{g} \cdot \mathtt{f}$.
A13 $(\mathtt{f} \cdot \mathtt{g}) \cdot \mathtt{h} = \mathtt{f} \cdot (\mathtt{g} \cdot \mathtt{h})$.
A14 $\mathtt{f} \cdot 1 = \mathtt{f}$.
A15 $\mathtt{f} \cdot (\mathtt{g} + \mathtt{h}) = (\mathtt{f} \cdot \mathtt{g}) + (\mathtt{f} \cdot \mathtt{h})$.
A16 $\mathtt{f} \geqslant \mathtt{g} \vee \mathtt{g} \geqslant \mathtt{f}$.
A17 $(\mathtt{f} \geqslant \mathtt{g} \wedge \mathtt{g} \geqslant \mathtt{h}) \rightarrow \mathtt{f} \geqslant \mathtt{h}$.
A18 $\mathtt{f} \geqslant \mathtt{g} \rightarrow \mathtt{f} + \mathtt{h} \geqslant \mathtt{g} + \mathtt{h}$.
A19 $(\mathtt{f} \geqslant \mathtt{g} \wedge \mathtt{h} > 0) \rightarrow \mathtt{f} \cdot \mathtt{h} \geqslant \mathtt{g} \cdot \mathtt{h}$.

5.4 Inference Rules

The only inference rule is modus ponens:

MP From ϕ and $\phi \rightarrow \psi$ derive ψ.

The notions of a proof, consistency, theorem etc are introduced in the usual way.

5.5 Completeness Theorem

It is easy to see that the introduced axiom system Ax_{PWF} is sound with respect to the class of measurable models. An immediate consequence of the A1 and MP is the following lemma:

Lemma 9. *Suppose that T is an arbitrary set of polynomial weight formulas. Then:*

1. *(L→) $T \vdash \phi \rightarrow \psi$ iff $T \cup \{\phi\} \vdash \psi$.*
2. *(L¬¬) $T \vdash \neg\neg\phi$ iff $T \vdash \phi$.*
3. *(L¬ →) $T \vdash \neg(\phi \rightarrow \psi)$ iff $T \vdash \phi$ and $T \vdash \neg\psi$.*
4. *(R¬¬) $T \cup \{\neg\neg\phi\} \vdash \psi$ iff $T \cup \{\phi\} \vdash \psi$.*

5. *(R¬ →)* $T \cup \{\neg(\phi \to \psi)\} \vdash \theta$ *iff* $T \cup \{\phi, \neg\psi\} \vdash \theta$.
6. *(R→)* $T \cup \{\phi \to \psi\} \vdash \theta$ *iff* $T \cup \{\neg\theta\} \vdash \phi$ *and* $T \cup \{\neg\theta\} \vdash \neg\psi$.

Theorem 10. *Every consistent set* $T \subseteq PWF$ *is satisfiable.*

Proof. Since the compactness theorem holds for the ${}^*\mathbb{R}$-valued polynomial weight formulas, it is sufficient to prove the simple completeness:

$$\vdash \phi \quad \text{iff} \quad \models \phi.$$

It is easy to prove that each theorem is a valid formula, so let us assume that ϕ is valid. By means of MP, A1, algebraic axioms and probabilistic axioms, we can transform ϕ into the formula ψ whose only connectives are negation and implication, and each polynomial weight term appearing in ψ is in the normal form. Clearly, ψ is also valid. Now applying Lemma 6 and the reduction strategy illustrated in the above examples, we obtain finitely many sequents of the form

$$\{\pm(\mathtt{f}_1 \geqslant 0), \cdots, \pm(\mathtt{f}_n \geqslant 0)\} \models \pm(\mathtt{g} \geqslant 0).$$

Since each of these sequents is satisfied, by A2 and Lemma 9, we may replace $\models$ by $\vdash$ in the entire reduction of ψ. It follows that ψ is a theorem, which concludes the proof. □

6 Conclusion

In this paper we proved compactness theorems for the logics with linear and polynomial weight formulas, introduced in [2], where the corresponding probability functions are ${}^*\mathbb{R}$-valued. In this way, the axiomatization for linear weight formulas given in [2] became complete in the extended sense. In addition, we provided a finitary extended-complete axiomatization for the polynomial weight formulas. As we noted above, [2] gave simple completeness for linear weight formulas, while failed to obtain a Hilbert-style complete axiomatization of polynomial weight formulas.

An obvious line of further research would be to try to apply the methodology of restoring compactness introduced here to some other probabilistic logics, e.g. the one introduced in [16] (logic with approximate conditional probabilities), which are used to model default reasoning. Alternatively, we might try adding the approximate equality $\approx$ to the present system Ax_{PWF}, in order to model defaults directly. However, it turns out that only the positive fragment of such logic might be amenable to our methodology and thus to recursive axiomatization, which remains to be further investigated.

References

1. Chang, C.C., Keisler, H.J.: Model theory, 3rd edn. North-Holland, Amsterdam (1990)
2. Fagin, R., Halpern, J., Megiddo, N.: A logic for reasoning about probabilities. Information and Computation 87(1–2), 78–128 (1990)

3. Flaminio, T.: Strong non-standard completeness for fuzzy logics. Soft Computing 12, 321–333 (2008)
4. Heifetz, A., Mongin, P.: Probability logic for type spaces. Games and economic behavior 35, 31–53 (2001)
5. Keisler, J.: Elementary calculus. An infinitesimal approach, 2nd edn., Boston, Massachusetts, Prindle, Weber and Schmidt (1986)
6. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55, 1–60 (1992)
7. Marchioni, E., Godo, L.: A Logic for Reasoning about Coherent Conditional Probability: A Modal Fuzzy Logic Approach. In: Alferes, J.J., Leite, J.A. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 213–225. Springer, Heidelberg (2004)
8. Nilsson, N.: Probabilistic logic. Artificial intelligence 28, 71–87 (1986)
9. Ognjanović, Z., Rašković, M.: A logic with higher order probabilities. In: Publications de l'institut mathematique, Nouvelle série, tome 60(74), pp. 1–4 (1996)
10. Ognjanović, Z., Rašković, M.: Some probability logics with new types of probability operators. J. Logic Computat. 9(2), 181–195 (1999)
11. Ognjanović, Z., Rašković, M.: Some first-order probability logics. Theoretical Computer Science 247(1–2), 191–212 (2000)
12. Ognjanović, Z., Marković, Z., Rašković, M.: Completeness Theorem for a Logic with imprecise and conditional probabilities. Publications de L'Institute Matematique (Beograd), ns. 78(92), 35–49 (2005)
13. Ognjanović, Z., Perović, A., Rašković, M.: Logic with the qualitative probability operator. Logic journal of IGPL doi:10.1093/jigpal/jzm031
14. Rašković, M.: Classical logic with some probability operators. In: Publications de l'institut mathematique, Nouvelle série, tome 53(67), pp. 1–3 (1993)
15. Rašković, M., Ognjanović, Z.: A first order probability logic LP_Q. In: Publications de l'institut mathematique, Nouvelle série, tome 65(79), pp. 1–7 (1999)
16. Rašković, M., Ognjanović, Z., Marković, Z.: A logic with Conditional Probabilities. In: Leite, J., Alferes, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 226–238. Springer, Heidelberg (2004)
17. Robinson, A.: Non-standard analysis. North-Holland, Amsterdam (1966)
18. van der Hoek, W.: Some considerations on the logic PeD: a logic combining modality and probability. Journal of Applied Non-Classical Logics 7(3), 287–307 (1997)

Combining Modes of Reasoning: An Application of Abstract Argumentation

Henry Prakken

Department of Information and Computing Sciences, Faculty of Science, Utrecht University & Faculty of Law, University of Groningen, The Netherlands

Abstract. Many reasoning problems involve subproblems that can be solved in different ways. Therefore, hybrid reasoning architectures have long been a research topic in AI. However, most work in this area has either focused on particular combinations of reasoning methods or has ignored the problem of handling alternative solutions to subproblems. The present paper proposes an abstract framework for combining modes of reasoning and handling alternative solutions. It is argued that current abstract argumentation systems are either too abstract or too specific for this purpose, so that an intermediate level of abstraction is needed.

1 Introduction

Many reasoning problems involve subproblems that can be solved in different ways. Legal reasoning provides a good example. First there is the problem of determining the facts. This problem can, for instance, be modelled with statistical methods [1], as abduction [2], as a problem of explanatory coherence [3] or as argumentation [4]. Then there is the problem of classifying the facts under the conditions of legal rules. This has been modelled, for instance, as case-based reasoning [5], as argumentation [6], and with neural nets [7]. Finally, when a rule's conditions have been satisfied, it must be applied. Since legal rules can have exceptions, this is often modelled as default reasoning [8].

Hybrid reasoning architectures have therefore long been a research topic in AI. However, most work in this area has either focused on particular combinations of formalisms, such as different logics or logics and probability theory (e.g. [2]), or on general methods for combining (monotonic) logics (e.g. [9]). The present paper instead proposes an abstract framework in which any set of problem solving methods can be combined, whether logical, probabilistic, connectionist or otherwise. Also, unlike the work on general techniques for combining logics, the aim is not to combine different methods into a single new one but to define a framework for defining intput-output relations between methods, while keeping the individual methods as they are. A final distinguishing feature of the present approach is the handling of *alternative* solutions to subproblems. A main motivation for this is the observation that in practical applications of KR & R formalisms often the hardest part is the *modelling* of a problem. While traditionally this is regarded as a knowledge engineering problem, in certain cases it is fruitful to explicitly model disagreements about the modelling of a problem.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 349–361, 2008.

I will in particular investigate the suitability of current abstract argumentation systems. The idea is that since these systems abstract from the nature of arguments, they could be suitable for combining different modes of reasoning. The most abstract argumentation system is that of Dung [10], which assumes nothing but a set of arguments with unspecified structure and an abstract relation of attack between arguments. However, this is too abstract for our purposes, since we also need support relations between arguments, to capture dependencies between subproblems. Abstract support relations have been added to [10]'s framework by [11] but this is still insufficient, since for giving guidance on how modes of reasoning can be combined, an account is needed of the nature of support and attack relations between modes of reasoning.

Still less abstract are systems with abstract inference rules, such as OSCAR [12], Abstract Argumentation Systems [13] and Carneades [14]. These systems model reasoning as the application of inference rules defined over some logical language. The systems are abstract in that nothing else is assumed on the nature of inference rules except that they are either strict (beyond attack) or defeasible (prone to attack). Support relations between arguments are captured in inference trees as usual in logic, and conflict relations are defined with logical negation.

At first sight, it would seem that the abstraction from specific sets of inference rules makes these systems suitable for combining modes of reasoning. However, it is not obvious that all modes of reasoning can be fruitfully modelled in the format of defeasible inference rules, let alone that their combination can be modelled as the combination of such rules: inference rules relate propositions, while we need to relate applications of possibly quite complex modes of reasoning.

Yet a strong point of rule-based argumentation systems is that they offer a well-founded formal machinery for managing conflicts given dependencies between issues. The present paper can be seen as an attempt to retain these strong points without having to commit to a modelling in terms of inference rules. This will be achieved by making the formalism isomorphic to the framework of [12]. Since that framework is known to be an instance of [10]'s abstract framework, the theory developed on abstract argumentation in the last thirteen years also applies to the present formal framework. The new contribution is that the elements of Pollock's framework will be given a different interpretation than in [12]: instead of propositions connected by inference rules, we will have applications of problem solving methods connected by output/input transformers.

Below in Section 2 the running example of this paper will be introduced, after which in section 3 the formalism is defined and illustrated with the running example. A discussion and concluding remarks are provided in Section 4.

2 A Running Example

Throughout this paper an imaginary legal case will be used as a running example. A criminal court has to decide whether a suspect is guilty of murder. The main rule of the criminal code is that someone who kills with intent is guilty of murder. This rule has two main statutory exceptions, namely, that there was

some ground of justification for the killing (for instance, self-defence) and that the killer cannot be held responsible for the killing (for instance, because he was insane). Therefore, application of the main rule can arguably best be modelled in a logic for reasoning with default rules. Let us, since it is so well known, use default logic as this logic. It divides input information into facts F, which are propositional or first-order formulas, and defaults D, which are domain-specific inference rules of the form $P{:}Q/R$, in which P, Q and R are propositional or first-order formulas. P is the prerequisite and R the consequent of the default, while Q is its justification. The informal meaning of a default is that R is derivable if P is derivable and Q is consistent with what is derivable. If defaults conflict, then the derivation process branches into alternative conclusion sets ('extensions'). The formal definition of extensions is rather involved but need not be explained here since our example is quite simple and intuitive. A formula is skeptically implied by a default theory if it is in all its extensions and it is credulously implied by the theory if it is in some but not all of its extensions. Exceptions to a default can be modelled as defaults whose consequents contradict its justification.

To apply the default theory to the issue of murder, information is needed about whether the suspect was the killer. The reasoning about this issue will be modelled as an application of Bayesian probability theory. The evidence is that the DNA of the suspect matches the DNA of the killer found at the crime scene. The conditional probability of a random match with that DNA given that a person is not the killer is given as 1 in 2 million. What we want to compute is the posterior conditional probability that the suspect is the killer given the match of his DNA with that of the killer.

This is not all, since to transform a posterior probability into an element of a default theory we need to apply proof standards. For simplicity this will be modelled as logical reasoning with a simple consistent first-order theory, saying that for criminal issues the proof standard is 0.99 (making precise 'beyond reasonable doubt') while for civil issues it is 0.51 (making precise 'on the balance of probabilities'). In fact there is an ongoing debate on how in legal applications of Bayesian statistics these legal proof standards should be defined [1]: such a debate could be modelled as a problem using some argumentation method.

We are still not done. Bayesian statistics requires prior probabilities to derive posterior probabilities from conditional ones and in our example there happens to be disagreement between the experts on what is the correct prior probability that the suspect is the killer. This disagreement will be modelled as an application of the method of first-order default logic (first-order since we want to talk about numerical values of variables). Discussions about prior probabilities are in fact very common in legal applications of Bayesian statistics [1]. Here the example will for reasons of space be kept very simple. The main disagreement is about the population of the potential suspects (let us assume that each of them has equal probability of being the murderer). One point of view is: 'the group of potential suspects consists of all male inhabitants over 16 of town X, which is 50.000 people', while another position is 'the group of potential suspects consists of all male inhabitants over 16 of quarter Y of town X, which is 10.000 people'.

With this information, it is convenient to use the odds formulation of Bayes' theorem:

$$\frac{\Pr(h|e)}{\Pr(\neg h|e)} = \frac{\Pr(h)}{\Pr(\neg h)} \times \frac{\Pr(e|h)}{\Pr(e|\neg h)}$$

Now we are interested in the prior odds that the suspect is the killer (k), in

$$\frac{\Pr(k)}{\Pr(\neg k)}$$

3 The Formal Framework

As illustrated by our running example, the main idea is that an overall problem is decomposed into several related subproblems, each solved in so-called problem treatments with possibly different methods. The input of a problem treatment is partly given and partly obtained from the output of one or more other problem treatments by so-called output-input transformers. A problem decomposition can also have conflict relations between its problem treatments, namely when they are alternative ways to model the same 'informal' problem, reflected by mutually contradictory input.

An important distinction of the framework is that between the object and metalanguages of problem solving methods. Different subproblems may be formulated in different object languages, such as propositional, first-order or modal logical languages, the language of probability theory, and so on. Yet the formalism defined below assumes that the input and output of all problems is described in a first-order language. This is since the output/input transformers operate on and produce first-order metalevel descriptions of the output and input of problem solving methods.

Let us now formalise these ideas.

Definition 1. *A* problem solving method M *is a tuple* $(\mathcal{L}^I_M, \mathcal{L}^O_M, R_M)$ *where*

- $\mathcal{L}^I_M$ *and* $\mathcal{L}^O_M$*, both first-order languages, are the input language and output language of* M*;*
- R_M *is a function from the powerset of* $\mathcal{L}^I_M$ *into the powerset of* $\mathcal{L}^O_M$*.*

The function R_M is the heart of a method M, specifying how M relates input to output. A method is applied in a 'problem treatment', which applies R_M to a set of problem input statements to produce a set of problem solutions:

Definition 2. *A* problem treatment *is a tuple* $\langle M, I^g, I^d, S\rangle$ *where*

- M *is a problem solving method;*
- $I^g \subseteq \mathcal{L}^I_M$ *is the given problem input;*
- $I^d \subseteq \mathcal{L}^I_M$ *is the derived problem input;*
- $I^g \cup I^d$ *is consistent;*
- $S \subseteq \mathcal{L}^O_M$ *are the problem solutions, such that* $S \subseteq R_M(I^g \cup I^d)$*.*

The elements of a problem treatment P will also be denoted by $M(P)$, $I^g(P)$, $I^d(P)$ and $S(P)$.

In applications of this definition the given input should include general constraints that ensure that the problem-specific input is of the required type. For example, if R is the reasoning of probability theory, then the constraints should ensure that the input is a probability distribution according to the laws of probability theory; this ensures, for instance, that two probability statements $\Pr(A) = x$ and $\Pr(A) = y$ (or $\Pr(A|B) = x$ and $\Pr(A|B) = y$) are inconsistent if $x \neq y$. In the rest of this paper such constraints will be left implicit if there is no danger for confusion.

Note that since the input language of a method is regarded as its metalanguage, consistency of a problem's input does not imply that it specifies a consistent theory. For example, the statement 'T contains p and $\neg p$' is a consistent first-order description of an inconsistent propositional theory T.

The four problem treatments of our running example are as follows. First the ultimate problem treatment P_1 on the issue of murder is given.

- $M(P_1)$ is propositional default logic (*MPropDL*) with the constraint that if $\varphi \in F$, then $\top : \varphi/\varphi \notin D$. It takes as input the specification of a propositional default theory $\Delta = (F, D)$.
- $I^g(P_1) =$
  $k \ \wedge \ i : \neg e_1 \ / \ m \in D$
  $k : \neg e_2 \ / \ i \in D$
  $(k \ \wedge \ i \ \wedge \ j) \supset e_1 \in F$
  $(k \ \wedge \ i \ \wedge \ \neg r) \supset e_1 \in F$
- We are interested in solutions that are ground instances of the formula 'm is x-implied by Δ', where x can take the values 'credulously', 'skeptically' or 'not', and Δ is the default theory specified by $I^g(P_1) \cup I^d(P_1)$.

The first default says that someone who kills (k) with intent (i) is guilty of murder (m) unless there is an exception to this statutory rule (e_1). The second default expresses the commonsense generalisation that killing is normally ($\neg e_2$) done with intent (for simplicity, exceptions to this rule are not listed). The facts in F contain the two exceptions to the rule on murder.

The problem treatment P_2 about the likelihood that the suspect is the killer is specified as follows.

- $M(P_2)$ is Bayesian probability theory (*MBayes*), taking as input probability distributions Π and sets E of evidence.
- $I^g(P_2) =$
  $E = \{d\}$
  $\Pr(d|k) = 1 \in \Pi$
  $\Pr(d|\neg k) = 0.0000005 \in \Pi$
- We are interested in solutions that are ground instances of the formula '$\Pr(k|E) \otimes p$ is implied by Π', where $\otimes$ ranges over $\{<, \leq, =, \geq, >\}$, and Π is the probability distribution specified by $I^g(P_2) \cup I^d(P_2)$.

Here d stands for 'DNA of the suspect matches DNA of the killer found at the crime scene' and k stands for 'The suspect is the killer'. The first conditional probability statement in $I^g(P_2)$ expresses that the DNA of the killer will certainly match with the DNA found at the murder scene, while the second one says that the probability of a random match of someone's DNA profile with any sample of DNA (so also with the DNA found at the murder scene) is one in two million.

The third problem treatment P_3 on what is the proof standard for the issue of killing is as follows.

- $M(P_3)$ = first-order predicate logic (*MFOL*), relating first-order theories T to output of the form 'φ is/is not implied by T'.
- $I^g(P_3) =$
  $\forall\varphi(\mathit{CriminalIssue}(\varphi) \supset \mathit{Standard}(\varphi) = 0.99) \in T$
  $\forall\varphi(\mathit{CivilIssue}(\varphi) \supset \mathit{Standard}(\varphi) = 0.51) \in T$
  $\mathit{CriminalIssue}(k) \in T$.
- We are interested in solutions that are ground instances of the formula '$\mathit{Standard}(k) = x$ is implied by T', where $0 \leq x \leq 1$, and T is the first-order theory specified by $I^g(P_3) \cup I^d(P_3)$.

The final problem treatment P_4 on what is the prior probability that the suspect is the killer is defined as follows.

- $M(P_4)$ is first-order default logic (*MFolDL*) with the constraint that if $\varphi \in F$, then $\top : \varphi/\varphi \notin D$, taking as input first-order default theories Δ and producing as output formulas of the form 'φ is x-implied by Δ' (where x can take the values 'credulously', 'skeptically' or 'not').
- $I^g(P_4) =$
  $\mathit{male16X}(k) : \neg e_1 \ / \ N = 50.000 \in D$
  $\mathit{male16YX}(k) : \neg e_2 \ / \ N = 10.000 \in D$
  $N = x : \neg e_3 \ / \ \mathit{priorodds}(k) = \frac{1}{x-1} \in D$
  $\mathit{male16X}(k) \in F$
  $\mathit{male16YX}(k) \in F$
- We are interested in solutions that are ground instances of the formula '$\mathit{priorodds}(k) = p$ is x-implied by Δ', where x can be 'credulously', 'skeptically' or 'not', and Δ is the default theory specified by $I^g(P_4) \cup I^d(P_4)$.

Note that the default theory of P_4 has two extensions, one containing $\mathit{priorodds}(k) = \frac{1}{9999}$ and the other containing $\mathit{priorodds}(k) = \frac{1}{49999}$. So our informal problem statement has two solutions, viz.

s_1 = '$\mathit{priorodds}(k) = \frac{1}{9999}$ is credulously implied by Δ'; and
s_2 ='$\mathit{priorodds}(k) = \frac{1}{49999}$ is credulously implied by Δ'.

Now the first solution yields (approximately) $\Pr(k|d) = 0.995$ while the second results in (approximately) $\Pr(k|d) = 0.976$, so since the proof standard for k is 0.99, the outcome of the debate on the prior odds of k is crucial to the question whether k can be proven beyond reasonable doubt.

As explained above, subproblems are related by output-input transformers. They are formally defined as follows.

Definition 3. *An* O/I transformer *from problem solving methods* $M_1, \ldots, M_m$ *to a problem solving method* M_n *is a function from* $\mathcal{L}^O_{M1} \times \ldots \times \mathcal{L}^O_{Mm}$ *into* $\mathcal{L}^I_{Mn}$.

In other words, an O/I transformer takes a set of formulas stated in the output languages of a set of problem solving methods, and converts it into a formula in the input language of a single problem solving method. Note that O/I transformers need not be translation functions between languages: firstly, an O/I transformer can take output formulated in a *set* of languages as input, and secondly, O/I transformers may express problem solving knowledge that goes beyond semantic knowledge, as the following transformers of our running example show.

We first define the following O/I transformer schemes from *MBayes* and *MFOL* to *MPropDL*.

> T_1: If $\Pr(H|E) \geq s$ is implied by Π in *MBayes* and 'The proof standard for H is s' is implied by T in *MFOL* then $H \in F$ in *MPropDL*.
> T_2: If $(1 - s) \leq \Pr(H|E) < s$ is implied by Π in *MBayes* and 'The proof standard for H is s' is implied by T in *MFOL* then $\top{:}\top/H \in D$ in *MPropDL*.

In other words, if a proof standard for H has been met, it is put into the facts F of a default theory, while if it has been met for neither H or $\neg H$ then both are put into the defaults D of the default theory (strictly speaking the defaults $\top{:}\top/H$ and $\top{:}\top/\neg H$ are added to D). With the laws of probability this implies for any h that $\top : h/h \in D$ iff $\top : \neg h/\neg h \in D$, that no proposition and its negation are in F and that for any proposition to which a transformer can be applied, either the proposition or its negation is in F, or one of the corresponding defaults is in D. Note that T_2 is an example of an O/I transformer that is not a translation function between languages: it takes input from a *set* of problems; and it expresses substantive legal instead of just semantic knowledge.

Which O/I transformers are appropriate from P_4 to P_2? Here there are several options. It is clear that any prior odds on k that is skeptically implied by a default theory will be part of Π in P_2. However, what if it is only credulously implied, as in the above default theory of P_4? Then there are two options. One is to 'block the ambiguity' and to input neither odds statement into P_2. In that case it is easy to see that no posterior probability on k can be calculated by P_2 so that P_1 receives no input on k and m is not derivable in P_1 in any sense. However, another option is to 'propagate the ambiguity', that is, to input two alternative prior odds of k into *MBayes* and to see whether their difference makes a difference in P_1. Here I do not want to argue for one option as the best and I simply choose the latter option to illustrate how the abstract formalism deals with conflicting problems. We thus have the following transformer scheme:

> T_3: If $priorodds(k) = p$ is skeptically or credulously implied by Δ in *MFolDL* then $priorodds(k) = p \in Pr$ in *MBayes*.

Problem treatments are assumed to be based on a so-called general problem specification, which is a set of pairs where each pair is a problem solving method plus a set of formulas in its input language.

Definition 4. *A* general problem specification *(GPS) is a set G of pairs (M, I^g) where*

1. *M is a problem solving method;*
2. *$I^g \subseteq \mathcal{L}^I_M$*
3. *If $(M, I^g) \in G$ and $(M', I^g) \in G$ then $M = M'$.*

Further constraints on GPS could be defined but their study must be left for future research.

In our running example we have in fact specified the following GPS:

$$\{(\mathit{MPropDL}, I^g(P_1)), (\mathit{MBayes}, I^g(P_2)), (\mathit{MFOL}, I^g(P_3)), (\mathit{MFolDL}, I^g(P_4))\}$$

It can now be defined how problem treatments can be combined.

Definition 5. *A* combination of problem treatments *based on a GPS G is a finite sequence $A = P_1, \ldots, P_m$ of problem treatments satisfying the following conditions for every P_i in A.*

1. *There exists a pair $(M, I^g) \in G$ such that $M(P_i) = M$ and $I^g(P_i) = I^g$; and*
2. *for each $\varphi_i \in I^d(P_i)$ there exist one or more problem treatments $P_j, \ldots, P_k$ $(j, k < i)$ in A with solutions $\varphi_j, \ldots, \varphi_k$ and a transformer T from $M(P_j) \times \ldots M(P_k)$ to $M(P_i)$ such that $T(\varphi_j, \ldots, \varphi_k) = \varphi_i$. (In such cases we say that P_i* depends *in A on $P_j, \ldots, P_k$.)*

For any combination of problem treatments $\mathcal{A}$ the set $\mathcal{A}^* = \{P \mid$ there exists an $A \in \mathcal{A}$ that contains $P\}$.

So the input of a problem treatment must be treated with the method specified by the GPS, and its 'derived' input must be provided via O/I transformers by, possibly combined, solutions of other problem treatments. Note that together Definitions 4 and 5 formalise that, for all problem treatments, I^g is given while I_d is obtained from other treatments.

On the basis of our example GPS the following combined problem treatments can be constructed for our solutions of interest.

$A_1 =$
$P_4 = \langle \mathit{MFolDL}, I^g(P_4), \emptyset, \{s_1, s_2\}\rangle$
$P_3 = \langle \mathit{MFOL}, I^g(P_3), \emptyset, \{s_3\}\rangle$
$P_{2a} = \langle \mathit{MBayes}, I^g(P_{2a}), \{s_1\}, \{s_4\}\rangle$ (T_3 applied to P_4)
$P_{1a} = \langle \mathit{MPropDL}, I^g(P_{1a}), \{s_5\}, \{s_6\}\rangle$ (T_1 applied to P_3 and P_{2a})

Where
$s_1 = \mathit{priorodds}(k) = \frac{1}{9999}$ is credulously implied by Δ
$s_2 = \mathit{priorodds}(k) = \frac{1}{49999}$ is credulously implied by Δ
$s_3 = \mathit{Standard}(k) = 0.99$ is implied by T
$s_4 = \Pr(k|E) = 0.995$ is implied by Π
$s_5 = k \in F$
$s_6 = m$ is skeptically implied by Δ

$A_2 =$
$P_4 = \langle \mathit{MFolDL}, I^g(P_4), \emptyset, \{s_1, s_2\}\rangle$
$P_3 = \langle \mathit{MFOL}, I^g(P_3), \emptyset, \{s_3\}\rangle$
$P_{2b} = \langle \mathit{MBayes}, I^g(P_{2b}), \{s_2\}, \{s_7\}\rangle$ (T_3 applied to P_4)
$P_{1b} = \langle \mathit{MPropDL}, I^g(P_{1b}), \{s_8\}, \{s_9, s_{10}\}\rangle$ (T_2 applied to P_3 and P_{2a})

Where
$s_7 = \Pr(k|E) = 0.976$ is implied by Π
$s_8 = k \in D$
$s_9 = m$ is credulously implied by Δ
$s_{10} = m$ is not skeptically implied by Δ

We now come to an important element of the formalism. To deal with situations in which different problem treatments provide mutually contradictory input for a certain problem, a binary relation of *conflict* between problem treatments is introduced. This relation captures situations in which the same 'informal' problem can be formalised in different ways. For instance, if an informal problem is to be modelled as an application of probability theory but there are two conflicting inputs on what are the correct probabilities for a certain set of probabilistic variables, then there should be two alternative probability distributions. To ensure this, the conflict relation is a necessary ingredient.

Definition 6. *A problem treatment* P conflicts *with a problem treatment* P' *if* $I^g(P) \cup I^d(P) \cup I^g(P') \cup I^d(P') \vdash \bot$. *A set* S *of problem treatments is called* conflict-free *if no member of* S *conflicts with a member of* S.

Note that the conflict relation is not only symmetric but also irreflexive since a problem treatment's input is assumed consistent.

In our example P_{2a} and P_{2b} conflict with each other since $I^d(P_{2a})$ contains '$\mathit{priorodds}(k) = \frac{1}{9999} \in \Pi$' while $I^d(P_{2b})$ contains '$\mathit{priorodds}(k) = \frac{1}{49999} \in \Pi$'. (Recall that $I^g(P_{2a})$ and $I^g(P_{2b})$ are assumed to contain axioms that make these two statements contradictory.) Likewise P_{1a} and P_{1b} are conflicting since $I^d(P_{1a})$ contains '$k \in F$' while $I^d(P_{1b})$ contains '$k \in D$'. The two combined problem treatments and these conflict relations are displayed in Figure 3 below (O/I transformers are displayed as solid lines and conflict relations as dashed lines.)

How can 'overall' solutions to general problem specifications be defined? Here we can exploit the fact that at a certain level of abstraction the structure of the present formalism is isomorphic to that of [12]. Firstly, in the present framework combined problem treatments are sequences of elementary problem treatments

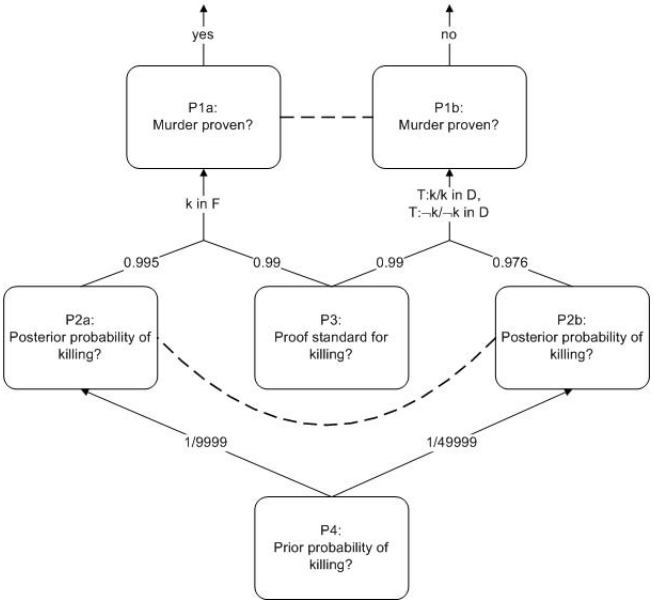


Fig. 1. The of the murder case

connected by applications of O/I transformers, so they can be represented as directed acyclic graphs. Likewise, in Pollock's framework arguments are sequences of elementary 'lines of argument' (statements plus a set of assumptions and a strength) connected by applications of inference rules, so Pollock's arguments can also be represented as directed acyclic graphs. Secondly, in the present framework conflict relations can hold between the elements of a combined problem treatment; likewise, in Pollock's framework 'defeat' relations can hold between lines of arguments. Now Pollock defines so-called defeat status assignments as labellings of argument lines, and he does so in terms of just a set of arguments represented as DAGs plus the defeat relation between lines of these arguments: he does not use the internal structure of argument lines nor the nature of the defeat relations for these purposes. Therefore, because of the just-explained isomorphism between the two frameworks, the same labellings can be defined for the present framework.

Note that despite this isomorphism, there are at a more concrete level two main differences between the present formalism and Pollock's. Firstly, while a problem treatment contains two sets of formulas related by an application of a problem solving method (which can be quite a complex process), a line of argument is just a single proposition plus a set of assumptions and a strength: within a line of argument no process takes place at all. Secondly, while problem treatments are connected via O/I transformers, lines of argument are connected by inference rules, which are quite different in nature from O/I transformers.

Let us now formalise these observations, adapting the notation and terminology of [12] to the present formalism.

Definition 7. *[labellings.] Let $\mathcal{A}$ be a combination of problem treatments. A* labelling *of $\mathcal{A}$ is a labelling of the elements of $\mathcal{A}^*$ defined as follows. For every $P \in \mathcal{A}^*$ belonging to the combination of problem treatments A:*

1. *P is labelled 'in' if:*
   (a) *all problem treatments on which P depends in A are labelled 'in'; and*
   (b) *all problems in $\mathcal{A}^*$ that conflict with P are labelled 'out';*
2. *P is labelled 'out' if:*
   (a) *some problem treatment on which P depends in A is labelled 'out'; or*
   (b) *P conflicts with a problem in $\mathcal{A}^*$ that is labelled 'in'.*

A labelling L of $\mathcal{A}$ is maximal if for all P and all labellings L' of $\mathcal{A}$ it holds that if P is in (out) in L' then P is in (out) in L. A labelling of $\mathcal{A}$ is complete if all nodes in $\mathcal{A}^*$ are labelled.

Intuitively, that a problem treatment is 'in' means that it can be regarded as a possible (but maybe not the only) way to solve a problem given the other problem treatments to which it is related by transformer and conflict relations.

In our running example the set $\{A_1, A_2\}$ has two complete labellings, viz L_1, in which P_{2b} and P_{1b} are out and the other problems are in, and L_2, in which P_{2a} and P_{1a} are out and the other problems are in.

It is now possible to define overall solutions to a GPS, analogously to the skeptical and credulous consequence notions of [12].

Definition 8. *Relative to a combination of problem treatments $\mathcal{A}$ a problem treatment $P \in \mathcal{A}^*$ is* defensible *if some maximal labelling of $\mathcal{A}$ labels P 'in', and P is* justified *if all maximal labellings of $\mathcal{A}$ label P 'in'. A formula φ is a* defensible solution *if φ is a solution of a defensible problem treatment, and φ is a* justified solution *if all labellings of $\mathcal{A}$ make some problem treatment 'in' that has φ as a solution.*

In our running example P_3 and P_4 are justified problem treatments while the other treatments are defensible. In consequence, there are two defensible solutions on the issue of murder. The first is that murder is skeptically derivable (based on L_1) while the second is that murder is not skeptically but only credulously derivable (based on L_2). So in the end there is no justification for convicting the suspect for murder.

The isomorphism with the formalism of [12] makes that the following properties can be proven.

Proposition 1. *Let $\mathcal{A}$ be an arbitrary combination of problem treatments.*

1. *All labellings of $\mathcal{A}$ are conflict-free.*
2. *$\mathcal{A}$ has at least one maximal labelling.*
3. *All maximal labellings of $\mathcal{A}$ are complete.*

4. *If $\mathcal{A}^*$ is conflict-free, then $\mathcal{A}$ has a unique labelling which is complete and which makes all problems of $\mathcal{A}^*$ 'in'.*

Proof. First, since Pollock's defeat status assignments are by Theorem 6.15 of [15] equivalent to an argumentation framework in the sense of [10] with preferred semantics, the results of [10] on preferred semantics also hold for labellings of GPS. Then (1) and (2) follow from Theorem 1 of [10]. Furthermore, (3) follows from Theorem 33 of [10] and symmetry and irreflexivity of the conflict relation between problems, so that all cycles through $\mathcal{C}$ relations are of even length. From this, property (4) also follows.

To conclude this section, it should be noted that Definitions 7 and 8 are not the only possible ones. Since [15] have shown that Pollock's system is an instance of Dung's abstract argumentation frameworks, any other semantics of [10] could also be used. The above definitions have been chosen since they allow for 'floating solutions' to a problem statement, analogous to floating conclusions in nonmonotonic logic. In the present context this seems useful: if a problem has alternative conflicting modellings but they solve some subproblem in the same way then the above definitions say that that solution is justified.

4 Discussion and Conclusion

This paper has presented an abstract formalism for combining different modes of reasoning. The model is abstract enough to include a wide variety of reasoning methods, ranging from purely symbolic to purely numeric ones. The aims to capture dependencies between problems and to manage alternative solutions of problems have been realised by exploiting a formal relation between our problem specifications and [12]'s system. Thus established theory on argumentation systems can be applied to a new phenomenon. A practical benefit of the present approach is the possibility to reason in one formalism about modelling decisions in another formalism, rather than to leave this to an unspecified knowledge engineering phase. Although several examples in this paper involved the combination of a nonmonotonic logic with probability theory, the formalism is by no means restricted to such combinations but can combine any set of reasoning methods. Moreover, in such combinations the individual methods can be left as they are, which is another practical benefit of the present approach.

Although the formalism uses techniques from the formal study of argumentation in AI, it does not require that problems are formalised in terms of inference rules, arguments and counterarguments as in [12,13,14]. Instead it allows the use of any reasoning method, which is achieved by abstracting from their internal structure. A key idea of this paper has been that O/I transformers do not operate on object-level descriptions of problems, as inference rules do, but on metalevel descriptions: thus existing formalisations (or even implementations) of problem solving methods can simply be 'plugged into' the present formalism, without the need to translate them into a new format. These ideas cannot be formalised with purely abstract support and conflict relations as in [11].

Finally, much work remains to be done. Firstly, it would be interesting to investigate whether conflicts between problem treatments can be resolved in other problem treatments (cf. [16] for an extension of [10]'s framework with attacks on attacks.) Also, a systematic study of the nature of O/I transformers between different modes of reasoning must be carried out, profiting whenever possible from existing work on integrating KR & R systems.

References

1. Lempert, R.: The new evidence scholarship: Analyzing the process of proof. Boston University Law Review 66, 439–477 (1986)
2. Poole, D.: Logical argumentation, abduction and Bayesian decision theory: a Bayesian approach to logical arguments and its application to legal evidential reasoning. Cardozo Law Review 22, 1733–1745 (2001)
3. Thagard, P.: Causal inference in legal decision making: Explanatory coherence vs. Bayesian networks. Applied Artificial Intelligence 18, 231–249 (2004)
4. Bex, F., Prakken, H., Reed, C., Walton, D.: Towards a formal account of reasoning about evidence: argumentation schemes and generalisations. Artificial Intelligence and Law 12, 125–165 (2003)
5. Ashley, K.: Modeling Legal Argument: Reasoning with Cases and Hypotheticals. MIT Press, Cambridge (1990)
6. Prakken, H., Sartor, G.: Modelling reasoning with precedents in a formal dialogue game. Artificial Intelligence and Law 6, 231–287 (1998)
7. Bench-Capon, T.: Neural networks and open texture. In: Proceedings of the Fourth International Conference on Artificial Intelligence and Law, pp. 292–297. ACM Press, New York (1993)
8. Prakken, H., Sartor, G.: A dialectical model of assessing conflicting arguments in legal reasoning. Artificial Intelligence and Law 4, 331–368 (1996)
9. Gabbay, D.: Fibring Logics. Oxford Logic Guides, vol. 38. Oxford University Press, Oxford (1999)
10. Dung, P.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n–person games. Artificial Intelligence 77, 321–357 (1995)
11. Cayrol, C., Lagasquie-Schiex, M.C.: Graduality in argumentation. Journal Artificial Intelligence Research 23, 245–297 (2005)
12. Pollock, J.: Cognitive Carpentry. A Blueprint for How to Build a Person. MIT Press, Cambridge (1995)
13. Vreeswijk, G.: Abstract argumentation systems. Artificial Intelligence 90, 225–279 (1997)
14. Gordon, T., Prakken, H., Walton, D.: The Carneades model of argument and burden of proof. Artificial Intelligence 171, 875–896 (2007)
15. Jakobovits, H., Vermeir, D.: Robust semantics for argumentation frameworks. Journal of Logic and Computation 9, 215–261 (1999)
16. Modgil, S.: An abstract theory of argumentation that accommodates defeasible reasoning about preferences. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 648–659. Springer, Heidelberg (2007)

Cheap Boolean Role Constructors for Description Logics*

Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler

Institute AIFB, Universität Karlsruhe, Germany
{sru,mak,phi}@aifb.uni-karlsruhe.de

Abstract. We investigate the possibility of incorporating Boolean role constructors on simple roles into some of today's most popular description logics, focussing on cases where those extensions do not increase complexity of reasoning. We show that the expressive DLs $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$, serving as the logical underpinning of OWL and the forthcoming OWL 2, can accommodate arbitrary Boolean expressions. The prominent OWL-fragment $\mathcal{SHIQ}$ can be safely extended by safe role expressions, and the tractable fragments $\mathcal{EL}^{++}$ and DLP retain tractability if extended by conjunction on roles, where in the case of DLP the restriction on role simplicity can even be discarded.

1 Introduction

Research on description logics (DLs) is directed by two main goals: increasing expressivity while preserving desirable computational properties such as decidability (as a factual *conditio sine qua non*) and efficiency of reasoning, the latter qualitatively estimated in terms of worst-case complexities. These antagonistic dimensions gave rise to a great variety of logics: $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$ being of high expressiveness and complexity represent one side of the spectrum, whereas the so called tractable fragments like $\mathcal{EL}^{++}$ and DLP provide lower expressivity yet allow for polynomial time reasoning.

In DL history, Boolean constructors (negation, conjunction, disjunction) on roles have occurred and have been investigated sporadically in many places, but have never been integrated into the mainstream of researched languages nor influenced standardisation efforts. In this paper, we argue that those constructors can – sometimes with appropriate restrictions – be incorporated into several of the most prominent DL languages, thereby significantly enhancing expressivity without increasing reasoning complexity.

To illustrate this gain in expressivity, we give some examples on the modelling capabilities of Boolean role constructors:

Universal role. A role U that connects all individuals of the described domain can e.g. be defined via $U \equiv \neg N$ as the complement of the empty role N, which in turn can be axiomatized by the GCI $\top \sqsubseteq \forall N.\bot$.

* Supported by the European Commission under contracts 027595 NeOn and 215040 ACTIVE, and by the Deutsche Forschungsgemeinschaft (DFG) under the ReaSem project.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 362–374, 2008.

Role conjunction. This modelling feature comes in handy if certain non-tree-like properties (namely cases where two individuals are interconnected by more than one role) have to be described. The fact that somebody testifying against a relative is not put under oath can e.g. be formalised by $\exists(\mathit{testifiesAgainst} \sqcap \mathit{relativeOf}).\top \sqsubseteq \neg \mathit{UnderOath}$. Likewise, role conjunction allows for specifying disjointness of roles, as $\mathsf{Dis}(R, S)$ can be paraphrased as $\top \sqsubseteq \forall(R \sqcap S).\bot$.

Concept products. Thoroughly treated in [1], the concept product statement $C \times D \sqsubseteq R$ expresses that any instance of C is connected with any instance of D via role R. As an example, the fact that alkaline solutions neutralise acid solutions, which could expressed by the concept product axiom $\mathit{AlkalineSolution} \times \mathit{AcidSolution} \sqsubseteq \mathit{neutralises}$, can equivalently be stated by $\mathit{AlkalineSolution} \sqsubseteq \forall(\neg \mathit{neutralises}).\neg \mathit{AcidSolution}$ by using role negation.

Qualified role inclusion. Likewise, the specialisation of roles due to concept memberships of the involved individuals can be expressed. The rule-like FOL statement $C(x) \wedge R(x, y) \wedge D(y) \rightarrow S(x, y)$ (expressing that any C-instance and D-instance that are interconnected by R are also interconnected by S) can be cast into the GCI $C \sqsubseteq \forall(R \sqcap \neg S).\neg D$. For example, the fact that any person of age having signed a contract which is legal is bound to that contract can be expressed by $\mathit{OfAge} \sqsubseteq \forall(\mathit{hasSigned} \sqcap \neg \mathit{boundTo}).\neg(\mathit{Contract} \sqcap \mathit{Legal})$.

The latter two types of statements have recently gained increased interest in the context of identifying rule-like fragments of DLs [2].

The rest of the paper is organised as follows. After providing the necessary definitions, we review existing work on Boolean role constructors. Then, we deal with the extension of $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$ by full Boolean role expressions on simple roles. Thereafter, we provide an according result for integrating safe Boolean role expressions into the description logic $\mathcal{SHIQ}$. The subsequent two sections settle the case for the tractable fragments $\mathcal{EL}^{++}$ and DLP, respectively, extending them by role conjunction. Finally, we conclude and elaborate on future work. Due to lack of space, some proofs had to be omitted. Those can be found in [3].

2 Preliminaries

In this section, we give the definition of the expressive description logic $\mathcal{SROIQB}_\mathrm{s}$ which is obtained from the well-known description logic $\mathcal{SROIQ}$ [4] by allowing arbitrary Boolean constructors on simple roles. We assume that the reader is familiar with description logics [5].

The DLs considered in this paper are based on four disjoint sets of *individual names* N_I, *concept names* N_C, and *simple role names* N_R^s (containing the *universal role* $U \in \mathsf{N}_R$) as well as *non-simple role names* N_R^n. Furthermore, we let $\mathsf{N}_R \coloneqq \mathsf{N}_R^\mathrm{s} \cup \mathsf{N}_R^\mathrm{n}$.

Definition 1. *A $\mathcal{SROIQB}_\mathrm{s}$ Rbox for N_R is based on a set* $\mathbf{R}$ *of* atomic roles *defined as* $\mathbf{R} \coloneqq \mathsf{N}_R \cup \{R^- \mid R \in \mathsf{N}_R\}$*, where we set* $\mathrm{Inv}(R) \coloneqq R^-$ *and* $\mathrm{Inv}(R^-) \coloneqq R$ *to simplify notation. In turn, we distinguish* simple atomic roles $\mathbf{R}^\mathrm{s} \coloneqq \mathsf{N}_R^\mathrm{s} \cup \mathrm{Inv}(\mathsf{N}_R^\mathrm{s})$ *and* non-simple atomic roles $\mathbf{R}^\mathrm{n} \coloneqq \mathsf{N}_R^\mathrm{n} \cup \mathrm{Inv}(\mathsf{N}_R^\mathrm{n})$.

In the sequel, we will use the symbols R, S, possibly with subscripts, to denote atomic roles.

The set of Boolean role expressions $\mathbf{B}$ *is defined as follows:*

$$\mathbf{B} ::= \mathbf{R} \mid \neg\mathbf{B} \mid \mathbf{B} \sqcap \mathbf{B} \mid \mathbf{B} \sqcup \mathbf{B}.$$

The set $\mathbf{B}_\mathrm{s}$ of simple *role expressions comprises all those role expressions containing only simple role names. In the sequel, V and W will denote simple role expressions if not stated otherwise. Moreover, a role expression will be called* safe*, if in its disjunctive normal form, every disjunct contains at least one non-negated role name.*

A generalised role inclusion axiom *(RIA) is a statement of the form $V \sqsubseteq W$ with simple role expressions V and W, or of the form*

$$S_1 \circ \ldots \circ S_n \sqsubseteq R$$

where each S_i is a simple role expression or a non-simple atomic role, and where R is a non-simple atomic role. A set of such RIAs will be called a generalised role hierarchy*. A role hierarchy is* regular *if there is a strict partial order $\prec$ on the non-simple roles $\mathbf{R}^\mathrm{n}$ such that*

- *$S \prec R$ iff $\mathrm{Inv}(S) \prec R$, and*
- *every RIA is of one of the forms*
  - *$R \circ R \sqsubseteq R$,*
  - *$R^- \sqsubseteq R$,*
  - *$S_1 \circ \ldots \circ S_n \sqsubseteq R$,*
  - *$R \circ S_1 \circ \ldots \circ S_n \sqsubseteq R$,*
  - *$S_1 \circ \ldots \circ S_n \circ R \sqsubseteq R$,*

  such that $R \in \mathsf{N}_R$ is a (non-inverse) role name, and $S_i \prec R$ for $i = 1, \ldots, n$ whenever S_i is non-simple.

A role assertion *is a statement of the form* $\mathsf{Ref}(R)$ *(*reflexivity*),* $\mathsf{Asy}(V)$ *(*asymmetry*), or* $\mathsf{Dis}(V, W)$ *(*role disjointness*), where V and W are simple role expressions, and R is a simple role expression or a non-simple role. A $\mathcal{SROIQB}_\mathrm{s}$* Rbox *is the union of a set of role assertions together with a role hierarchy. A $\mathcal{SROIQB}_\mathrm{s}$ Rbox is regular if its role hierarchy is regular.*

Definition 2. *Given a $\mathcal{SROIQB}_\mathrm{s}$ Rbox $\mathcal{R}$, the set of* concept expressions $\mathbf{C}$ *is defined as follows:*

- *$\mathsf{N}_C \subseteq \mathbf{C}$, $\top \in \mathbf{C}$, $\bot \in \mathbf{C}$,*
- *if $C, D \in \mathbf{C}$, $R \in \mathbf{B}_\mathrm{s} \cup \mathbf{R}^\mathrm{n}$ a simple role expression or non-simple role, $V \in \mathbf{B}_\mathrm{s}$ a simple role expression, $a \in \mathsf{N}_I$, and n a non-negative integer, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\{a\}$, $\forall R.C$, $\exists R.C$, $\exists V.\mathsf{Self}$, $\leq n\, V.C$, and $\geq n\, V.C$ are also concept expressions.*

Throughout this paper, the symbols C, D will be used to denote concept expressions. A $\mathcal{SROIQB}_\mathrm{s}$ Tbox *is a set of* general concept inclusion axioms *(GCIs) of the form $C \sqsubseteq D$.*

An individual assertion *can have any of the following forms: $C(a)$, $R(a, b)$, $\neg V(a, b)$, $a \not\approx b$, with $a, b \in \mathsf{N}_I$ individual names, $C \in \mathbf{C}$ a concept expression, and $R, S \in \mathbf{R}$ roles with S simple. A $\mathcal{SROIQB}_\mathrm{s}$* Abox *is a set of individual assertions.*

Name	Syntax	Semantics
inverse role	R^-	$\{(x,y) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid (y,x) \in R^\mathcal{I}\}$
universal role	U	$\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
role negation	$\neg V$	$\{(x,y) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid (x,y) \notin R^\mathcal{I}\}$
role conjunction	$V \sqcap W$	$V^\mathcal{I} \cap W^\mathcal{I}$
role disjunction	$V \sqcup W$	$V^\mathcal{I} \cup W^\mathcal{I}$
top	$\top$	$\Delta^\mathcal{I}$
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
nominals	$\{a\}$	$\{a^\mathcal{I}\}$
univ. restriction	$\forall R.C$	$\{x \in \Delta^\mathcal{I} \mid (x,y) \in R^\mathcal{I}$ implies $y \in C^\mathcal{I}\}$
exist. restriction	$\exists R.C$	$\{x \in \Delta^\mathcal{I} \mid$ for some $y \in \Delta^\mathcal{I}$, $(x,y) \in R^\mathcal{I}$ and $y \in C^\mathcal{I}\}$
Self concept	$\exists V.\mathsf{Self}$	$\{x \in \Delta^\mathcal{I} \mid (x,x) \in V^\mathcal{I}\}$
qualified number restriction	$\leq n\, V.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid (x,y) \in V^\mathcal{I}$ and $y \in C^\mathcal{I}\} \leq n\}$
	$\geq n\, V.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid (x,y) \in V^\mathcal{I}$ and $y \in C^\mathcal{I}\} \geq n\}$

Fig. 1. Semantics of concept and role constructors in $\mathcal{SROIQB}_\mathrm{s}$ for an interpretation $\mathcal{I}$ with domain $\Delta^\mathcal{I}$

A $\mathcal{SROIQB}_\mathrm{s}$ knowledge base KB *is the union of a regular Rbox* $\mathcal{R}$*, and an Abox* $\mathcal{A}$ *and Tbox* $\mathcal{T}$ *for* $\mathcal{R}$*. We use the term* axiom *to uniformly refer to any single statement contained in* $\mathcal{R}$*,* $\mathcal{A}$*, or* $\mathcal{T}$*.*

We further give the semantics of $\mathcal{SROIQB}_\mathrm{s}$ knowledge bases.

Definition 3. *An interpretation* $\mathcal{I}$ *consists of a set* $\Delta^\mathcal{I}$ *called* domain *together with a function* $\cdot^\mathcal{I}$ *mapping individual names to elements of* $\Delta^\mathcal{I}$*, concept names to subsets of* $\Delta^\mathcal{I}$*, and role expressions to subsets of* $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$*.*

The function $\cdot^\mathcal{I}$ *is inductively extended to role and concept expressions as shown in Fig. 1. An interpretation* $\mathcal{I}$ satisfies *an axiom* φ *if we find that* $\mathcal{I} \models \varphi$*:*

- $\mathcal{I} \models V \sqsubseteq W$ *if* $V^\mathcal{I} \subseteq W^\mathcal{I}$*,*
- $\mathcal{I} \models V_1 \circ \ldots \circ V_n \sqsubseteq R$ *if* $V_1^\mathcal{I} \circ \ldots \circ V_n^\mathcal{I} \sqsubseteq R^\mathcal{I}$ *(*$\circ$ *being overloaded to denote the standard composition of binary relations here),*
- $\mathcal{I} \models \mathsf{Ref}(R)$ *if* $R^\mathcal{I}$ *is a reflexive relation,*
- $\mathcal{I} \models \mathsf{Asy}(V)$ *if* $V^\mathcal{I}$ *is antisymmetric and irreflexive,*
- $\mathcal{I} \models \mathsf{Dis}(V,W)$ *if* $V^\mathcal{I}$ *and* $W^\mathcal{I}$ *are disjoint,*
- $\mathcal{I} \models C \sqsubseteq D$ *if* $C^\mathcal{I} \subseteq D^\mathcal{I}$*,*
- $\mathcal{I} \models C(a)$ *if* $a^\mathcal{I} \in C^\mathcal{I}$*,*
- $\mathcal{I} \models R(a,b)$ *if* $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$*,*
- $\mathcal{I} \models \neg V(a,b)$ *if* $(a^\mathcal{I}, b^\mathcal{I}) \notin V^\mathcal{I}$*,*
- $\mathcal{I} \models a \not\approx b$ *if* $a^\mathcal{I} \neq b^\mathcal{I}$*.*

An interpretation $\mathcal{I}$ satisfies *a knowledge base* KB *(we then also say that* $\mathcal{I}$ *is a* model *of* KB *and write* $\mathcal{I} \models$ KB*) if it satisfies all axioms of* KB*. A knowledge base* KB *is* satisfiable *if it has a model. Two knowledge bases are* equivalent *if they have exactly*

the same models, and they are equisatisfiable *if either both are unsatisfiable or both are satisfiable.*

We obtain $\mathcal{SROIQ}$ from $\mathcal{SROIQB}_s$ by disallowing all junctors in role expressions. Further details on $\mathcal{SROIQ}$ can be found in [4]. We have omitted here several syntactic constructs that can be expressed indirectly, especially role assertions for transitivity, reflexivity of simple roles, and symmetry. Moreover, the DL $\mathcal{SHOIQ}$ is obtained from $\mathcal{SROIQ}$ by discarding the universal role as well as reflexivity, asymmetry, role disjointness statements and allowing only RIAs of the form $R \sqsubseteq S$ or $R \circ R \sqsubseteq R$.

3 Related Work

Boolean constructors on roles have been investigated in the context of both description and modal logics. [6] used them extensively for the definition of a DL that is equivalent to the two-variable fragment of FOL.

As a classical result on complexities, it was shown in [7], that augmenting $\mathcal{ALC}$ with full Boolean role constructors ($\mathcal{ALCB}$) leads to NExpTime-completeness of the standard reasoning tasks (while restricting to role negation [7] or role conjunction [8] only retains ExpTime-completeness). This complexity does not further increase when allowing for inverses, qualified number restrictions, and nominals as was shown in [8] by a polynomial translation of $\mathcal{ALCIQB}$ into C^2, the two variable fragment of first order logic with counting quantifiers, which in turn was proven to be NExpTime-complete in [9]. Also the recently considered description logic $\mathcal{ALBO}$ [10] falls in that range of NExpTime-complete DLs.

On the contrary, it was also shown in [8] that restricting to *safe* Boolean role constructors keeps $\mathcal{ALC}$'s reasoning complexity in ExpTime, even when adding inverses and qualified number restrictions ($\mathcal{ALCIQb}$).

For logics including modelling constructs that deal with role concatenation like transitivity or – more general – complex role inclusion axioms, results on complexities in the presence of Boolean role constructors are more sparse. [11] shows that $\mathcal{ALC}$ can be extended by negation and regular expressions on roles while keeping reasoning within ExpTime. Furthermore, [12] provided ExpTime complexity for a similar logic that includes inverses and qualified number restriction but reverts to safe negation on roles.

An extension of $\mathcal{SHIQ}$ with role conjunction (denoted $\mathcal{SHIQ}^{\sqcap}$) is presented in [13] in the context of conjunctive query answering, the results implying an upper bound of 2ExpTime.

4 $\mathcal{SROIQB}_s$ and $\mathcal{SHOIQB}_s$

In this section, we show that adding arbitrary (i.e. also unsafe) Boolean role expressions to the widely known description logics $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$ does not harm their reasoning complexities – N2ExpTime [14] and NExpTime [8], respectively – if this extension is restricted to simple roles.

Note that in the sequel, $\mathcal{SHOIQ}$ (resp. $\mathcal{SHOIQB}_s$) will be treated as a special case of $\mathcal{SROIQ}$ (resp. $\mathcal{SROIQB}_s$), as most considerations hold for both cases.

Table 1. Additional transformation for $\mathcal{SROIQB}_s$ and $\mathcal{SHOIQB}_s$. A, B are concept names. V, W are simple role expressions. V_i are simple role expressions or non-simple roles. $\hat{V}$ is a simple role expression that is not just a role. R is a non-simple role name. $\overline{S}$ is a new simple role name.

$A \sqsubseteq \forall \hat{V}.B$	$\mapsto$	$\{A \sqsubseteq \forall \overline{S}.B, \hat{V} \sqsubseteq \overline{S}\}$
$A \sqsubseteq \geq n\, \hat{V}.B$	$\mapsto$	$\{A \sqsubseteq \geq n\, \overline{S}.B, \overline{S} \sqsubseteq \hat{V}\}$
$A \sqsubseteq \leq n\, \hat{V}.B$	$\mapsto$	$\{A \sqsubseteq \leq n\, \overline{S}.B, \hat{V} \sqsubseteq \overline{S}\}$
$A \sqsubseteq \exists \hat{V}.\mathsf{Self}$	$\mapsto$	$\{A \sqsubseteq \exists \overline{S}.\mathsf{Self}, \overline{S} \sqsubseteq \hat{V}\}$
$\mathsf{Dis}(V, W)$	$\mapsto$	$\{V \sqcap W \sqsubseteq \overline{S}, \top \sqsubseteq \forall \overline{S}.\bot\}$
$V_1 \circ \ldots \circ \hat{V} \circ \ldots \circ V_n \sqsubseteq R$	$\mapsto$	$\{V_1 \circ \ldots \circ \overline{S} \circ \ldots \circ V_n \sqsubseteq R, \hat{V} \sqsubseteq \overline{S}\}$

As shown in [14], any $\mathcal{SROIQ}$ ($\mathcal{SHOIQ}$) knowledge base can be transformed into an equisatisfiable knowledge base containing only axioms of the form:

$$\begin{array}{lll} A \sqsubseteq \forall R.B & \bigsqcap A_i \sqsubseteq \bigsqcup B_j & S_1 \sqsubseteq S_2 \\ A \sqsubseteq \geq n\, S.B & A \equiv \{a\} & S_1 \sqsubseteq S_2^- \\ A \sqsubseteq \leq n\, S.B & A \equiv \exists S.\mathsf{Self} & \mathsf{Dis}(S_1, S_2) \\ & R_1 \circ \ldots \circ R_n \sqsubseteq R. & \end{array}$$

Trivially, this normalization can be applied to $\mathcal{SROIQB}_s$ ($\mathcal{SHOIQB}_s$) as well, yielding the same types of axioms whereas simple role expressions may occur in the place of simple roles. A second transformation carried out by exhaustively applying the transformation steps depicted in Table 1 yields an equisatisfiable knowledge base containing only the original axiom types depicted above (i.e. again only simple role names in places of $S_{(i)}$ and role names in places of R_i) and just one additional axiom type $W \sqsubseteq V$ with W,V simple role expressions. As shown in [14], any of these original axiom types except the one containing role concatenation can be translated into C^2, the two-variable fragment of first order logic with counting quantifiers. The additionally introduced type of axiom can clearly also be transformed into C^2 statements namely into the proposition $\forall xy(\Phi(W) \rightarrow \Phi(V))$ where Φ is inductively defined by:

$$\begin{aligned} \Phi(S) &= S(x, y) \\ \Phi(S^-) &= S(y, x) \\ \Phi(\neg V) &= \neg \Phi(V) \\ \Phi(V \sqcap W) &= \Phi(V) \wedge \Phi(W) \\ \Phi(V \sqcup W) &= \Phi(V) \vee \Phi(W) \end{aligned}$$

Further following the argumentation from [14], the remaining complex role inclusions not directly convertible into C^2 can be taken into account by cautiously materializing the consequences resulting from their interplay with axioms of the type $A \sqsubseteq \forall R.B$ through automata encoding techniques – see also [15]. This way, one obtains a C^2 theory that is satisfiable exactly if the original knowledge base is. In the case of $\mathcal{SROIQ}$ (and hence $\mathcal{SROIQB}_s$), this can result in an exponential blowup of the knowledge base while for $\mathcal{SHOIQB}_s$ (and hence $\mathcal{SHOIQ}$) the transformation is time polynomial. Thus we see that the upper complexity bounds for $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$ carry over to $\mathcal{SROIQB}_s$

and $\mathcal{SHOIQB}_\text{s}$ by just a slight extension of the according proofs from [14] while the lower bounds follow directly from those of $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$. Hence, we can establish the following theorem.

Theorem 1. *Knowledge base satisfiability checking, instance retrieval, and computing class subsumptions for $\mathcal{SROIQB}_\text{s}$ ($\mathcal{SHOIQB}_\text{s}$) knowledge bases is* N2ExpTime*-complete (*NExpTime*-complete).*

While the results established in this section are rather straightforward consequences of known results, their implications for practice might be more significant: they show that the DLs underlying OWL and OWL 2 can be extended by arbitrary Boolean constructors on simple roles without increasing the worst case complexity of reasoning.

5 $\mathcal{SHIQb}_\text{s}$

$\mathcal{SHIQ}$ is a rather expressive fragment obtained from $\mathcal{SHOIQ}$ by disallowing nominals, where (in contrast to the NExpTime-complete $\mathcal{SHOIQ}$) reasoning is known to be ExpTime-complete [8].

In this section we will introduce the extension of $\mathcal{SHIQ}$ by safe role expressions on simple roles. Thereafter, we will present a technique for removing transitivity statements from $\mathcal{SHIQb}_\text{s}$ knowledge bases in a satisfiability preserving way. This yields two results: on the one hand, we provide a way how existing reasoning procedures for $\mathcal{ALCHIQb}$ like e.g. those described in [8,12,16] can be used to solve $\mathcal{SHIQb}_\text{s}$ reasoning tasks. On the other hand, as the transformation procedure can be done in polynomial time, the known upper bound for the complexity of reasoning in $\mathcal{ALCHIQb}$ – namely ExpTime – carries over to $\mathcal{SHIQb}_\text{s}$.

Definition 4. *A $\mathcal{SHIQb}_\text{s}$ knowledge base is a $\mathcal{SHOIQB}_\text{s}$ knowledge base that contains no nominals and only safe role expressions.*

Based on a fixed knowledge base KB*, we define $\sqsubseteq^*$ as the smallest binary relation on the non-simple atomic roles $\mathbf{R}_\text{n}$ such that:*

- *$R \sqsubseteq^* R$ for every atomic role R,*
- *$R \sqsubseteq^* S$ and $\text{Inv}(R) \sqsubseteq^* \text{Inv}(S)$ for every Rbox axiom $R \sqsubseteq S$, and*
- *$R \sqsubseteq^* T$ whenever $R \sqsubseteq^* S$ and $S \sqsubseteq^* T$.*

Given an atomic non-simple role R, we write $\mathsf{Trans}(R) \in \text{KB}$ as an abbreviation for: $R \circ R \sqsubseteq R \in \text{KB}$ or $\text{Inv}(R) \circ \text{Inv}(R) \sqsubseteq \text{Inv}(R) \in \text{KB}$.

Slightly generalising according results from [8,17] (as we allow safe boolean expressions – in GCIs and role inclusion axioms – already for the original logic), we now show that any $\mathcal{SHIQb}_\text{s}$ knowledge base can be transformed into an equisatisfiable knowledge base not containing transitivity statements.

Definition 5. *Given a $\mathcal{SHIQb}_\text{s}$ knowledge base* KB*, let $\mathsf{clos}(\text{KB})$ denote the smallest set of concept expressions where*

- *$\mathsf{NNF}(\neg C \sqcup D) \in \mathsf{clos}(\text{KB})$ for any Tbox axiom $C \sqsubseteq D$,*
- *$D \in \mathsf{clos}(\text{KB})$ for every subexpression D of some concept $C \in \mathsf{clos}(\text{KB})$,*

- $\mathsf{NNF}(\neg C) \in \mathsf{clos}(\mathrm{KB})$ *for any* $\leq n\, R.C \in \mathsf{clos}(\mathrm{KB})$,
- $\forall S.C \in \mathsf{clos}(\mathrm{KB})$ *whenever* $\mathsf{Trans}(S) \in \mathrm{KB}$ *and* $S \sqsubseteq^* R$ *for a role* R *with* $\forall R.C \in \mathsf{clos}(\mathrm{KB})$.

Moreover, let $\Omega(\mathrm{KB})$ *denote the knowledge base obtained from* KB *by*

- *Removing all transitivity axioms* $R \circ R \sqsubseteq R$ *and*
- *Adding the axiom* $\forall R.C \sqsubseteq \forall S.(\forall S.C)$ *for every* $\forall R.C \in \mathsf{clos}(\mathrm{KB})$ *with* $\mathsf{Trans}(S) \in \mathrm{KB}$ *and* $S \sqsubseteq^* R$.

Proposition 1. *Let* KB *be a* $\mathcal{SHIQ}b_s$ *knowledge base. Then,* KB *and* $\Omega(\mathrm{KB})$ *are equisatisfiable.*

Taking into account that the presented transformation is time polynomial, this result can now be employed to determine the complexity of $\mathcal{SHIQ}b_s$.

Theorem 2. *Knowledge base satisfiability checking, instance retrieval, and computing class subsumptions for* $\mathcal{SHIQ}b_s$ *knowledge bases is* ExpTime*-complete.*

Proof. Clearly, all standard reasoning problems can be reduced to knowledge base satisfiability checking as usual.

Now, by Proposition 1, any given $\mathcal{SHIQ}b_s$ knowledge base KB can be transformed into an $\mathcal{ALCHIQ}b$ knowledge base $\Omega(\mathrm{KB})$ in polynomial time. Furthermore, all role inclusion axioms can be removed from $\Omega(\mathrm{KB})$ as follows. First, all role names contained in $\Omega(\mathrm{KB})$ can be declared to be simple without violating the syntactic constraints. Second, every role inclusion axiom $V \sqsubseteq W$ (with V, W being safe by definition) can be equivalently transformed into the GCI $\top \sqsubseteq \forall(V \sqcap \neg W).\bot$. Note that then $V \sqcap \neg W$ is safe as well and therefore admissible. Moreover the transformation is obviously time linear. So we end up with an $\mathcal{ALCIQ}b$ knowledge base whose satisfiability checking is ExpTime-complete due to [8]. □

So we have shown that allowing safe Boolean expressions on simple roles does not increase the ExpTime reasoning complexity of $\mathcal{SHIQ}$. On the other hand, the recent results on $\mathcal{SHIQ}^{\sqcap}$ [13] seem to indicate that the role simplicity condition is essential for staying within ExpTime even though no definite hardness result for general $\mathcal{SHIQ}^{\sqcap}$ was provided. The safety condition on role expressions, in turn, is clearly needed: dropping it would lead to a DL comprising $\mathcal{ALCB}$ which is known to be NExpTime-complete [7].

6 $\mathcal{EL}^{++}(\sqcap_s)$

In this section, we investigate role conjunction for the DL $\mathcal{EL}^{++}$ [18], for which many typical inference problems can be solved in polynomial time. We simplify our presentation by omitting concrete domains from $\mathcal{EL}^{++}$ – they are not affected by our extension and can be treated as shown in [18].

Definition 6. *An atomic role of* $\mathcal{EL}^{++}(\sqcap_s)$ *is a (non-inverse) role name. An* $\mathcal{EL}^{++}(\sqcap_s)$ *role expression is a simple role expression that contains only role conjunctions. An* $\mathcal{EL}^{++}(\sqcap_s)$ *Rbox is a set of generalised role inclusion axioms (using* $\mathcal{EL}^{++}(\sqcap_s)$ *role expressions and non-simple atomic roles). An* $\mathcal{EL}^{++}(\sqcap_s)$ *Tbox is a* $\mathcal{SROIQB}_s$ *Tbox that contains only the concept constructors:* $\sqcap$, $\exists$, $\top$, $\bot$ *and only* $\mathcal{EL}^{++}(\sqcap_s)$ *role expressions.*

Note that we do not have any requirement for regularity of roles but we have to introduce the notion of role simplicity in the context of $\mathcal{EL}^{++}$. In a first step, we observe that any $\mathcal{EL}^{++}(\sqcap_s)$ knowledge base can be converted into a normal form.

Definition 7. *An $\mathcal{EL}^{++}(\sqcap_s)$ knowledge base* KB *is in* normal form *if it contains only axioms of one of the following forms:*

$$\begin{array}{rcl} A & \sqsubseteq & C \\ \exists R.A & \sqsubseteq & B \end{array} \qquad \begin{array}{rcl} A \sqcap B & \sqsubseteq & C \\ A & \sqsubseteq & \exists R.B \\ R \sqcap S & \sqsubseteq & T \end{array} \qquad \begin{array}{rcl} R & \sqsubseteq & T \\ R \circ S & \sqsubseteq & T \end{array}$$

where $A, B \in \mathsf{N}_C \cup \{\{a\} \mid a \in \mathsf{N}_I\} \cup \{\top\}$, $C \in \mathsf{N}_C \cup \{\{a\} \mid a \in \mathsf{N}_I\} \cup \{\bot\}$, and $R, S, T \in \mathsf{N}_R$.

Proposition 2. *Any $\mathcal{EL}^{++}(\sqcap_s)$ knowledge base can be transformed into an equisatisfiable $\mathcal{EL}^{++}(\sqcap_s)$ knowledge base in normal form. The transformation can be done in linear time.*

Subsequently, we show that the only axiom type of this normal form not covered by $\mathcal{EL}^{++}$ can be removed from an $\mathcal{EL}^{++}(\sqcap_s)$ knowledge base while preserving satisfiability if the relevant consequences are materialized before.

Definition 8. *Given an $\mathcal{EL}^{++}(\sqcap_s)$ knowledge base* KB *in normal form, let $\Theta^{\sqcap}$(KB) denote the knowledge base obtained from* KB *by*

- *Adding $R_1 \sqsubseteq R_2$ for all $R_2 \in R_1^{\sqsubseteq}$ where $S^{\sqsubseteq} \subseteq \mathsf{N}_R$ denotes the smallest set of role names containing S and satisfying*
  - *$T \in S^{\sqsubseteq}$, whenever $R \in S^{\sqsubseteq}$ and $R \sqsubseteq T \in$ KB as well as*
  - *$T \in S^{\sqsubseteq}$, whenever $R_1, R_2 \in S^{\sqsubseteq}$ and $R_1 \sqcap R_2 \sqsubseteq T \in$ KB,*
- *Removing every axiom of the form $S_1 \sqcap S_2 \sqsubseteq R$ and instead adding the axioms $\exists S_1.\{o\} \sqcap \exists S_2.\{o\} \sqsubseteq \exists R.\{o\}$ for every individual name $\{o\}$.*

Note that $\Theta^{\sqcap}$(KB) can be computed in polynomial time. In particular, finding the closed sets $R^{\sqsubseteq}$ can be done in linear time w.r.t. the size of KB, e.g. using the *linclosure* algorithm from [19].

Proposition 3. *Let* KB *be an $\mathcal{EL}^{++}(\sqcap_s)$ knowledge base. Then,* KB *and $\Theta^{\sqcap}$(KB) are equisatisfiable.*

The shown reduction – besides providing a way of using existing $\mathcal{EL}^{++}$ reasoning algorithms for reasoning in $\mathcal{EL}^{++}(\sqcap_s)$ – now gives rise to the complexity result for $\mathcal{EL}^{++}(\sqcap_s)$.

Theorem 3. *Knowledge base satisfiability checking, instance retrieval, and computing class subsumptions for $\mathcal{EL}^{++}(\sqcap_s)$ knowledge bases is* P*-complete w.r.t. the size of the knowledge base.*

Proof. Given an arbitrary $\mathcal{EL}^{++}(\sqcap_s)$ knowledge base KB, Proposition 2 ensures that it can be transformed in polynomial time into an equisatisfiable knowledge base KB′ in normal form. Again in polynomial time, we can compute the knowledge base $\Theta^{\sqcap}$(KB′)

that – by Proposition 3 – is equisatisfiable with KB′ (and hence also with KB). Finally, as $\Theta^{\sqcap}\mathrm{KB}'$ is an $\mathcal{EL}^{++}$ knowledge base, we can check satisfiability in polynomial time.

P-hardness for $\mathcal{EL}^{++}(\sqcap_s)$ follows from the well known P-hardness of $\mathcal{EL}$ (also being a straightforward consequence of the P-completeness of Horn satisfiability). □

We finish this section with some general remarks. On the one hand, note that conjunction on roles enhances expressivity of $\mathcal{EL}^{++}$ significantly. For example, it allows for the following modelling features:

- Disjointness of two simple roles S, R. This feature, also provided by $\mathcal{SROIQ}$ as $\mathsf{Dis}(S, R)$, can be modelled in $\mathcal{EL}^{++}(\sqcap_s)$ by the axiom $\exists(S \sqcap R).\top \sqsubseteq \bot$.
- Atleast cardinality constraints on the right hand side of a GCI. The axiom $A \sqsubseteq \geq n\, R.B$ can be modelled by the axiom set $\{R_i \sqsubseteq R, A \sqsubseteq \exists R_i.B \mid 1 \leq i \leq n\} \cup \{\exists(R_i \sqcap R_j).\top \sqsubseteq \bot \mid 1 \leq i < j \leq n\}$ where $R_1, \ldots, R_n$ are new simple role names.

On the other hand, it is easy to see that incorporating more than just conjunction on simple roles into $\mathcal{EL}^{++}$ would render the respective fragment intractable at best: Allowing conjunction on non-simple roles would even lead to undecidability as stated in Theorem 1 of [20]. Allowing disjunction or negation on simple roles would allow to model disjunction on concepts: for instance, the GCI $A \sqsubseteq B \sqcup C$ can be expressed by the axiom set $\{A \sqsubseteq \exists(R \sqcup S).\top, \exists R.\top \sqsubseteq B, \exists S.\top \sqsubseteq C\}$ or the axiom set $\{A \sqcap \exists R.\{o\} \sqsubseteq C, A \sqcap \exists \neg R.\{o\} \sqsubseteq B\}$ for new roles R, S and a new individual name o. Hence, any extension of $\mathcal{EL}^{++}$ into this direction would be ExpTime-hard [18].

7 DLP(⊓)

Description Logic Programs (DLP) constitutes a tractable knowledge representation formalism in the spirit of (Horn) logic programming [21]. Essentially, it consists of those $\mathcal{SHOIQ}$ axioms which can be naively translated into (non-disjunctive) Datalog, such that the original knowledge base and its translation are semantically equivalent. As such it represents the fragment of $\mathcal{SHOIQ}$ that can entail neither disjunctive information nor the existence of anonymous individuals as extensively studied in the context of Horn description logics [22]. Though rather complex syntactic definitions can be given to characterise all admissible axioms of such logics, we use a simpler definition comprising all essential expressive features of DLP without including all their syntactic varieties.

Definition 9. *Atomic roles of DLP are defined as in $\mathcal{SROIQ}$, including inverse roles. A* DLP body concept *is any $\mathcal{SROIQ}$ concept expression that includes only concept names, nominals, $\sqcap$, $\exists$, $\top$, and $\bot$. A* DLP head concept *is any $\mathcal{SROIQ}$ concept expression that includes only concept names, nominals, $\sqcap$, $\forall$, $\top$, $\bot$, and expressions of the form $\leq 1.C$ where C is a DLP body concept.*

A DLP knowledge base *is a set of Rbox axioms of the form $R \sqsubseteq S$ and $R \circ R \sqsubseteq R$, Tbox axioms of the form $C \sqsubseteq D$, and Abox axioms of the form $D(a)$ and $R(a, b)$, where $C \in \mathbf{C}$ is a body concept, $D \in \mathbf{C}$ is a head concept, and $a, b \in \mathsf{N}_I$ are individual names.*

DLP(⊓) *knowledge bases are defined just as DLP knowledge bases, with the addition that conjunctions of roles may occur in DLP(⊓) in all places where roles occur in DLP.*

Note that we do not have to distinguish between simple and non-simple roles for DLP.

In [22] it is shown that DLP is of polynomial worst-case complexity. This can be seen most easily by realising that DLP knowledge bases can be transformed in polynomial time (in the size of the knowledge base) into an equisatisfiable set of function-free first-order Horn rules (i.e. non-disjunctive Datalog rules) with at most three variables per formula. On the basis of this result, it is easy to show that DLP($\sqcap$) is also of polynomial complexity. We give a brief account of the argument.

Consider a DLP($\sqcap$) knowledge base K. We now perform the following transformation of K: For any role conjunction $R_1 \sqcap \cdots \sqcap R_n$ occurring in the knowledge base, replace the conjunction by a new role R, and add the axioms $R_1 \sqcap \cdots \sqcap R_n \sqsubseteq R$ and $R \sqsubseteq R_i$, for all $i = 1, \ldots, n$, to the knowledge base.

The resulting knowledge base is obviously equisatisfiable with K. It consists of two types of axioms: Axioms which are in DLP and axioms of the form $R_1 \sqcap \cdots \sqcap R_n \sqsubseteq R$. The latter axioms correspond to function-free Horn rules with only two variables. Hence, any DLP($\sqcap$) knowledge base can be transformed in polynomial time into an equisatisfiable set of function-free Horn rules.

Theorem 4. *Knowledge base satisfiability checking, instance retrieval, and computing class subsumptions for DLP($\sqcap$) knowledge bases is* P*-complete w.r.t. the size of the knowledge base.*

Proof. First note that instance retrieval and class subsumption can be reduced to satisfiability checking: Retrieval of instances for a class C is done by checking for all individuals a if they are in C – which in turn is reduced to satisfiability checking by adding the axioms $C \sqcap E \sqsubseteq \bot$ and $E(a)$ to the knowledge base, where E is a new atomic class name. Class subsumption $C \sqsubseteq D$ is reduced by adding the axioms $C(a)$, $E(a)$ and $D \sqcap E \sqsubseteq \bot$, for a new individual a and a new atomic class name E.

Now to check satisfiability of a DLP($\sqcap$) knowledge base, it is first transformed into an equisatisfiable set of function-free first-order Horn rules as mentioned above. The satisfiability of such a set of formulae can be checked in polynomial time, since any Horn logic program is semantically equivalent to its *grounding* (the set of all possible ground instances of the given rules based on the occurring individual names). For a program with a bounded number n of variables per rule, this grounding is bounded by $r \times i^n$, where i is the number of individual names and r is the number of rules in the program. Finally, the evaluation of ground Horn logic programs is known to be in P.

P-hardness for DLP directly follows from the P-completeness of satisfiability checking for sets of propositional Horn clauses. □

8 Conclusion

In our work, we have thoroughly investigated the reasoning complexities of DLs allowing for Boolean constructors on simple roles. We found that the expressive DLs $\mathcal{SROIQ}$ (being the basis of the forthcoming OWL 2 standard) and $\mathcal{SHOIQ}$ (the logical underpinning of OWL) can accommodate full Boolean role operators while keeping their reasoning complexities N2ExpTime and NExpTime, respectively. Likewise, the

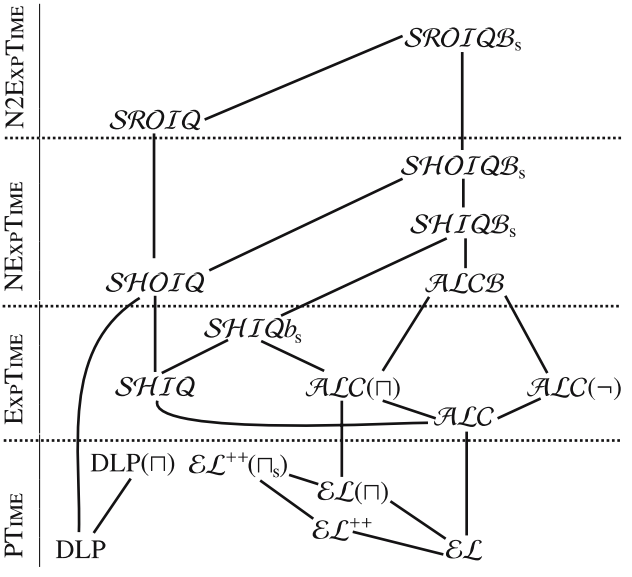


Fig. 2. Overview of complexities and expressivity relationships of DLs in the context of this paper

ExpTime-complete $\mathcal{SHIQ}$ can be safely extended by safe Boolean expressions. Finally, both the tractable fragments $\mathcal{EL}^{++}$ and DLP retain polynomial time reasoning complexity when adding just role conjunction, where in the case of DLP the role simplicity condition is not necessary. Figure 2 shows our findings integrated with other well-known complexity results relevant in this respect.

In particular, we want to draw the reader's attention to the fact that – as opposed to hitherto proposed ways – the modelling of concept products and qualified role inclusions as presented in Section 1 does *not* automatically render the inferred roles non-simple. Moreover, due to the safety of the respective axiom, qualified role inclusions can even be modelled in $\mathcal{SHIQb}_s$.

Future work on that topic includes the further integration of the established results with our work on DL Rules [2], as well as the further investigation of the effects on complexity and decidability when allowing for Boolean constructors on non-simple roles.

Finally note that our results for $\mathcal{SHIQb}_s$, $\mathcal{EL}^{++}(\sqcap_s)$, and DLP($\sqcap$) provide direct ways for adapting existing reasoning algorithms for $\mathcal{SHIQ}$, $\mathcal{EL}^{++}$, and DLP, respectively. For $\mathcal{SROIQB}_s$ and $\mathcal{SHOIQB}_s$, however, setting up efficient algorithms seems less straightforward and represents another interesting direction of future research.

References

1. Rudolph, S., Krötzsch, M., Hitzler, P.: All elephants are bigger than all mice. In: 21st Int. Workshop on Description Logics (DL 2008) (2008)
2. Krötzsch, M., Rudolph, S., Hitzler, P.: Description logic rules. In: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008). IOS Press, Amsterdam (2008)

3. Krötzsch, M., Rudolph, S., Hitzler, P.: Cheap boolean role constructors for description logics. Technical report, Universität Karlsruhe (TH) (2008), http://www.aifb.uni-karlsruhe.de/WBS/sru/TR-RKH-bool-role.pdf
4. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press, Menlo Park (2006)
5. Baader, F., et al. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2007)
6. Borgida, A.: On the relative expressiveness of description logics and predicate logics. Artif. Intell. 82, 353–367 (1996)
7. Lutz, C., Sattler, U.: The complexity of reasoning with boolean modal logics. In: Wolter, F., Wansing, H., de Rijke, M., Zakharyaschev, M. (eds.) Advances in Modal Logics, vol. 3. CSLI Publications, Stanford (2001)
8. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. PhD thesis, RWTH Aachen, Germany (2001)
9. Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. Journal of Logic, Language and Information 14(3), 369–395 (2005)
10. Schmidt, R.A., Tishkovsky, D.: Using tableau to decide expressive description logics with role negation. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 438–451. Springer, Heidelberg (2007)
11. Lutz, C., Walther, D.: PDL with negation of atomic programs. Journal of Applied Non-Classical Logics 15, 189–213 (2005)
12. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics: An automata-theoretic approach. In: AAAI, pp. 391–396. AAAI Press, Menlo Park (2007)
13. Glimm, B., Lutz, C., Horrocks, I., Sattler, U.: Answering conjunctive queries in the SHIQ description logic. Journal of Artificial Intelligence Research 31, 150–197 (2008)
14. Kazakov, Y.: SRIQ and SROIQ are harder than SHOIQ. In: Proc. 21st Int. Workshop on Description Logics (DL 2008), CEUR WS Proceedings (2008)
15. Demri, S., Nivelle, H.: Deciding regular grammar logics with converse through first-order logic. J. of Logic, Lang. and Inf. 14, 289–329 (2005)
16. Rudolph, S., Krötzsch, M., Hitzler, P.: Terminological reasoning in SHIQ with ordered binary decision diagrams. In: Proceedings of the 23rd AAAI Conference on Artficial Intelligence (AAAI 2008). AAAI Press, Menlo Park (2008)
17. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, Universität Karlsruhe (TH), Germany (2006)
18. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), Edinburgh, UK. Morgan-Kaufmann Publishers, San Francisco (2005)
19. Maier, D.: The Theory of Relational Databases. Computer Science Press (1983)
20. Krötzsch, M., Rudolph, S., Hitzler, P.: Conjunctive queries for a tractable fragment of OWL 1.1. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 310–323. Springer, Heidelberg (2007)
21. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proc. 12th Int. Conf. on World Wide Web (WWW 2003), pp. 48–57. ACM Press, New York (2003)
22. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for Horn description logics. In: Proc. 22nd AAAI Conf (AAAI 2007). AAAI Press, Menlo Park (2007)

Improved Second-Order Quantifier Elimination in Modal Logic

Renate A. Schmidt

School of Computer Science, The University of Manchester

Abstract. This paper introduces improvements for second-order quantifier elimination methods based on Ackermann's Lemma and investigates their application in modal correspondence theory. In particular, we define refined calculi and procedures for solving the problem of eliminating quantified propositional symbols from modal formulae. We prove correctness results and use the approach to compute first-order frame correspondence properties for modal axioms and modal rules. Our approach can solve two new classes of formulae which have wider scope than existing classes known to be solvable by second-order quantifier elimination methods.

1 Introduction

Propositional modal logics, when defined syntactically, are a priory second-order logics, but can often be characterized by classes of model structures (frames) which satisfy first-order conditions. Using second-order quantifier elimination methods, these first-order conditions, called frame correspondence properties, can frequently be derived from the axioms in an axiomatization of a logic. For example, the axiom $\mathbf{D} = \forall p[\Box p \rightarrow \Diamond p]$ is translated to $\forall P \forall x[\forall y[R(x,y) \rightarrow P(y)] \rightarrow \exists z[R(x,z) \wedge P(z)]]$. Eliminating the second-order quantifier $\forall P$ with a second-order quantifier elimination method returns the formula $\forall x \exists y[R(x,y)]$. This is the first-order frame correspondence property equivalent to the axiom $\mathbf{D}$. Rather than translating the modal input formula into second-order logic and then passing it to a second-order quantifier elimination algorithm, we focus on approaches that perform second-order quantifier elimination directly in modal logic. Only in a subsequent step translation to first-order logic is performed. Given $\forall p[\Box p \rightarrow \Diamond p]$ such an approach eliminates $\forall p$ and returns the formula $\Diamond \top$. Subsequently this is translated to first-order logic to give $\forall x \exists y[R(x,y)]$.

Several automated second-order quantifier elimination methods exist (and have various applications [7]). These methods belong to two categories: (i) those based on Ackermann's Lemma [5,3], and (ii) those based on resolution [6,2]. In this paper we are interested in methods of the first kind. Ackermann's Lemma [1] tells us when quantified predicate variables are eliminable from second-order formulae. Common to methods based on Ackermann's Lemma are inference rules, called *Ackermann rules*, which are designed to apply to formulae in forms specified by the lemma. Crucial are therefore suitable equivalence preserving

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 375–388, 2008.

transformation rules which transform formulae into appropriate forms so that Ackermann rules can be applied. Past experience shows that optimization techniques which simplify formulae are crucial in making it easier, and even possible, to obtain the necessary syntactic forms for Ackermann rules to become applicable. While it has been possible to show strong completeness and canonicity results [3,4], developing computationally feasible transformation and optimization techniques has so far received little attention.

In this paper we present an enhanced approach based on Ackermann's Lemma designed for propositional multi-modal tense logics. Which symbols are to be eliminated can be flexibly specified. The quantified propositional variables to be eliminated are the *non-base symbols*. All other symbols are assumed to be *base symbols*. Given a set of modal logic formulae, the goal is to find a set of formulae equivalent to the original set but does not contain any of the non-base symbols.

A main design criterion of our approach has been increased efficiency, as well as increased success rate and scope. In our approach an ordering on the non-base symbols must be specified. The ordering provides a mechanism for controlling how a derivation is constructed, reducing non-determinism and reducing the search space. An important novelty is the inclusion of general notion of redundancy, which allows for the flexible definition of practical simplification and optimization techniques to reduce the search space further and improve the success rate.

A second motivation is to gain a better understanding of when quantifier elimination methods succeed, and to pinpoint precisely which techniques are crucial for successful termination. We define two new classes of formulae for which our approach succeeds: the class $\mathcal{C}$ and an hierarchical version called $\mathcal{C}^>$. The classes define normal forms for when Ackermann-based second-order quantifier elimination methods succeed and subsume both the well-known Sahlqvist class of formulae [9] and also the class of monadic inductive formulae due to Goranko and Vakarelov [8]. We present minimal requirements for successful termination for all these classes. This allows us to sharpen and strengthen existing results on the termination behaviour of second-order quantifier elimination methods.

The third motivation is the provision of methods for correspondence theory of *modal rules*. Previous work has only investigated the utilization of second-order quantifier elimination methods for computing correspondence properties of *modal axioms*. We show that the methods can be used for the study of correspondence properties of modal rules as well.

Our approach is closely related to the DLS algorithm of [5] and the SQEMA algorithm of [3], but we introduce various improvements which can be transferred to both these algorithms. Both SQEMA and DLS are based on versions of Ackermann's Lemma. They both operate on formulae in negation normal form. Because negation normal form has a number of drawbacks, we use a different normal form. Our normal form allows our method to succeed quicker and more often because more cases of obvious redundancies can be detected and eliminated with little effort. Another difference is that our calculi contain less non-deterministic

choices, and are equipped with general, efficient redundancy criteria which are important for the success rate and practical implementations.

The paper is structured as follows. The next section defines the necessary logical apparatus and notation. The calculi which form the basis of our approach are introduced in Section 3. Section 4 describes two procedures based on these calculi. In Section 5 it is described how these can be used to compute first-order frame correspondence properties for modal axioms and rules, and examples are given. Section 6 introduces the two new classes $\mathcal{C}$ and $\mathcal{C}^{>}$. $\mathcal{C}$ is basically the subclass of formulae in $\mathcal{C}^{>}$ for which the order in which non-base symbols are eliminated, sign switching and redundancy elimination are not essential. It is shown that the Sahlqvist class of formulae and the class of monadic inductive formulae are subsumed by $\mathcal{C}$ and thus also $\mathcal{C}^{>}$.

The paper is based on an unpublished manuscript by the author from 2006 and Chapter 13 in [7]. Due to space limitations many details and all proofs are omitted. These can be found in the long version at `http://www.cs.man.ac.uk/~schmidt/publications/Schmidt08b.pdf`, together with more examples.

2 Modal Tense Logics

The general setting of this paper is the logic $\mathbf{K}^n_{(m)}(\breve{}, \pi+)$ with forward and backward looking modalities, nominals, and second-order quantification over propositional variables. $\mathbf{K}^n_{(m)}(\breve{}, \pi+)$ is the extension of the *multi-modal tense logic* $\mathbf{K}_{(m)}(\breve{})$ with second-order quantification and nominals.

Let V be an enumerable set of propositional variables $p_1, p_2, \ldots$ and let A be an enumerable set of nominals $a_1, a_2, \ldots$. Intuitively, nominals are propositional variables which are true in exactly one world. A formula in $\mathbf{K}^n_{(m)}(\breve{}, \pi+)$ is either a propositional atom, i.e., a propositional variable, $\bot$ or a nominal, or a formula of the form $\neg\alpha$, $\alpha \wedge \beta$, $\Box_k\alpha$, $\Box^{\breve{}}_k\alpha$, and $\forall p_i[\alpha]$, where α and β denote $\mathbf{K}^n_{(m)}(\breve{}, \pi+)$-formulae, $i \geq 1$ and $k \geq 1$. The connectives $\top$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\Diamond_k$, $\Diamond^{\breve{}}_k$, $\exists p_i$ are defined as usual, e.g., the converse diamond operator is specified by $\Diamond^{\breve{}}_i\alpha =^{df} \neg\Box^{\breve{}}_k\neg\alpha$, where α denotes an arbitrary $\mathbf{K}^n_{(m)}(\breve{}, \pi+)$-formula. We assume that $\vee$ and $\wedge$ are commutative and associative.

We say $\Box^{\breve{}}_k$ (resp. $\Diamond^{\breve{}}_k$, $\Box_k$, $\Diamond_k$) is the *converse operator* of $\Box_k$ (resp. $\Diamond_k$, $\Box^{\breve{}}_k$, $\Diamond^{\breve{}}_k$). As alternative notation we also use $\Box_{\overline{k}}$ for $\Box^{\breve{}}_k$. To simplify the notation we use the symbol κ for k or $\overline{k}$. If $\kappa = k$ ($\kappa = \overline{k}$) then $\Box^\kappa$ denotes $\Box_k$ ($\Box^{\breve{}}_\kappa$). Further let $\Box^{\kappa,\breve{}}$ denote the converse of $\Box^\kappa$. ($\Diamond^\kappa$, $\Diamond^{\kappa,\breve{}}$ are defined similarly.) Let $R^\kappa(s,t)$ be $R_k(s,t)$, if $\kappa = k$, and $R_k(t,s)$, if $\kappa = \overline{k}$, for any terms s and t.

Now we define the semantics of $\mathbf{K}^n_{(m)}(\breve{}, \pi+)$. A Kripke frame is a relational structure $\langle W, \langle R_k\rangle_k\rangle$, where W is a non-empty set (of worlds) and $\langle R_k\rangle_k$ is a family of binary (accessibility) relations over W. A Kripke model (interpretation) is a tuple $M = \langle W, \langle R_k\rangle_k, v\rangle$, where $\langle W, \langle R_k\rangle_k\rangle$ is the underlying frame and v is the valuation function. v assigns subsets of W to propositional variables.

In modal logic there are various ways of defining the semantics of second-order quantification. We use the standard definition in which the semantics of quantified propositional variables is defined in terms of p-equivalent models.

Let p be a propositional symbol and let M and M' be two Kripke models. We say M and M' are *p-equivalent* if M and M' coincide but differ possibly in the valuation of p. More generally, suppose $\Sigma = \{p_1, \ldots, p_m\} \subseteq V$, M and M' are *Σ-equivalent* if M and M' are the same but differ possibly in the valuation of the propositional symbols in Σ.

Truth of arbitrary $\mathbf{K}^n_{(m)}(^\smile, \pi+)$-formulae in a world x of a model M is defined inductively by:

$$M, x \models \top; \qquad M, x \models p_i \text{ iff } x \in v(p_i); \qquad M, x \models a_j \text{ iff } v(a_j) = \{x\};$$
$$M, x \models \alpha \wedge \beta \text{ iff both } M, x \models \alpha \text{ and } M, x \models \beta; \quad M, x \models \neg\alpha \text{ iff } M, x \not\models \alpha;$$
$$M, x \models \Box^\kappa \alpha \text{ iff for any } y \in W,\ R^\kappa(x, y) \Rightarrow M, y \models \alpha;$$
$$M, x \models \forall p_i[\alpha] \text{ iff for any model } M' \ p_i\text{-equivalent to } M,\ M', x \models \alpha.$$

If $M, x \models \alpha$ for some x then α is said to be *locally satisfiable* in M. We write $M \models \alpha$ when $M, x \models \alpha$ for all worlds x of M.

$\mathbf{K}^n_{(m)}(^\smile, \pi+)$ can be embedded into second-order logic with equality and constant symbols using the standard translation mapping. The *standard translation* of formulae is a mapping π inductively defined as follows:

$$\pi(\top, x) = \top; \qquad \pi(p_i, x) = P_i(x); \qquad \pi(a_j, x) = x \approx c_j;$$
$$\pi(\alpha \wedge \beta, x) = \pi(\alpha, x) \wedge \pi(\beta, x); \qquad \pi(\neg\alpha, x) = \neg\pi(\alpha, x);$$
$$\pi(\Box^\kappa \alpha, x) = \forall y[R^\kappa(x, y) \rightarrow \pi(\alpha, y)]; \qquad \pi(\forall p_i[\alpha], x) = \forall P_i[\pi(\alpha, x)].$$

Here, x denotes a first-order variable and y is a fresh first-order variable, whenever required. It is assumed that the symbols P_i are predicate symbols uniquely associated with the propositional variables p_i. The nominals a_j are uniquely associated with constants c_j, and R_k is a binary predicate symbol representing the accessibility relation associated with $\Box_k$ and $\Box_k^\smile$. The symbol $\approx$ denotes the first-order equality predicate. (We use the symbol $=$ for syntactic equality.)

We have that α is locally satisfiable in $\mathbf{K}^n_{(m)}(^\smile, \pi+)$ iff $\exists x[\pi(\alpha, x)]$ is satisfiable in classical second-order logic. (Note the freely occurring propositional symbols in α are interpreted as propositional constants.)

Axiomatizations of traditional modal logics without second-order quantifiers can be defined in terms of modal axioms and rules. In this paper a *modal rule* is a pair Δ/Δ' of sets of modal formulae without second-order quantifiers. A *modal axiom* is a modal rule in which Δ is empty. The semantics of a rule is the following: For any model M,

$$\forall \overline{p}\big[M \models \textstyle\bigwedge_{\alpha\in\Delta} \alpha \Rightarrow M \models \bigwedge_{\beta\in\Delta'} \beta\big],$$

where $\overline{p}$ denotes the sequence of propositional variables occurring in Δ and Δ'. Thus, the semantics of an axiom $\alpha = \bigwedge \Delta'$ is given by $\forall \overline{p}[M \models \alpha]$, or $M \models \forall \overline{p}[\alpha]$.

Accordingly, the translation of modal rules and axioms is defined by a mapping Π from rules/axioms to second-order formulae, given by:

$$\Pi(\Delta/\Delta') = \forall \overline{P}\big[\forall x[\pi(\textstyle\bigwedge_{\alpha\in\Delta} \alpha, x) \rightarrow \forall x[\pi(\bigwedge_{\beta\in\Delta'} \beta, x)]\big];$$

$$\Pi(\alpha) = \forall \overline{P} \forall x[\pi(\alpha, x)] \quad (\equiv \forall x[\pi(\forall \overline{p}[\alpha], x)]).$$

Suppose $\mathbf{L}$ is a sublogic of $\mathbf{K}^n_{(m)}(^\smile, \pi+)$. By $\mathbf{L}(\mathcal{R})$ we denote the extension of $\mathbf{L}$ with a set $\mathcal{R}$ of modal rules or axioms. We have that a formula α is locally satisfiable in $\mathbf{L}(\mathcal{R})$ iff $\bigwedge_{R \in \mathcal{R}} \Pi(R) \wedge \exists x[\pi(\alpha, x)]$ is satisfiable in second-order logic. (As above the freely occurring propositional symbols in α are interpreted as constants.)

For the rest of the paper we need some more definitions, terminology and notation. By ϵ we denote the empty sequence, by . sequence concatenation, and by σ any (possibly empty) sequence of natural numbers which may be overlined. By definition, let $\Box^\epsilon \alpha =^{df} \alpha$, $\Box^{\epsilon,\smile} \alpha =^{df} \alpha$, $\Box^{\kappa.\sigma} \alpha =^{df} \Box^\kappa \Box^\sigma \alpha$, and $\Box^{\kappa.\sigma,\smile} \alpha =^{df} \Box^{\sigma,\smile} \Box^{\kappa,\smile} \alpha$. Define $\Diamond^\epsilon$, $\Diamond^\sigma$, $\Diamond^{\sigma,\smile}$ similarly. For any terms s and t, let $R^\epsilon(s,t) =^{df} s \approx t$, $R^{\epsilon,\smile}(s,t) =^{df} s \approx t$, $R^{\kappa.\sigma}(s,t) =^{df} \exists x[R^\kappa(s,x) \wedge R^\sigma(x,t)]$, and $R^{\kappa.\sigma,\smile}(s,t) =^{df} \exists x[R^\sigma(t,x) \wedge R^\kappa(x,s)]$.

For any formula α, not containing $\sim$, let $\sim\alpha$ denote β, if $\alpha = \neg\beta$, and $\neg\alpha$, otherwise.

A *modal atom* is a propositional atom or a formula $\Box^\kappa \beta$, where β is in modal disjunctive normal form. A *modal literal* is a modal atom or a negated modal atom. A formula α is in *modal disjunctive normal form* iff it is a disjunction of conjunctions of modal literals. Every $\mathbf{K}^n_{(m)}(^\smile)$-formula can be effectively reduced to modal disjunctive normal form.

An occurrence of a propositional variable in a modal formula α is *positive (negative)* iff it occurs in the scope of an even (odd) number of explicit and implicit negations. A modal formula α is *positive (negative) in a variable* p iff all occurrences of p in α are positive (negative). Let Σ denote a (possibly empty) set of propositional variables. A modal formula α is *positive* (*negative*) (in Σ) iff all occurrences of p (from Σ) in α are positive (negative).

If $\Sigma = \{p_1, \ldots, p_m\} \subseteq V$ then $\exists\Sigma[\alpha]$, resp. $\forall\Sigma[\alpha]$, denotes $\exists p_1 \ldots \exists p_m[\alpha]$, resp. $\forall p_1 \ldots \forall p_m[\alpha]$.

We write $\beta(p)$ to indicate that p occurs freely in the formula β. By $\beta^{\gamma_1,\ldots\gamma_n}_{\alpha_1,\ldots\alpha_n}$ we mean the formula obtained from β by replacing all occurrences of γ_i by α_i. If N denotes a set of formulae, then $N^{\gamma_1,\ldots\gamma_n}_{\alpha_1,\ldots\alpha_n}$ is defined similarly.

3 Modal Ackermann Calculi

The rules of our calculi operate on sets of $\mathbf{K}^n_{(m)}(^\smile)$-clauses. A *(modal) clause* C is a disjunction of modal logic formulae which is globally satisfiable, i.e., true in every world of a model. For any inference rule of the form N/N', where N and N' are sets of $\mathbf{K}^n_{(m)}(^\smile)$-clauses, their meaning is characterized by the following property: $M \models \exists\Sigma[\bigwedge N]$ iff $M \models \exists\Sigma[\bigwedge N']$, for any $\mathbf{K}^n_{(m)}(^\smile, \pi+)$-model M.

The calculi are parameterized by a set $\Sigma = \{p_1, \ldots, p_m\} \subseteq V$ of propositional variables, an ordering $>$ on Σ and an ordering $\succ$ of formulae and clauses. The symbols in Σ are referred to as the *non-base symbols*. All other symbols are assumed to be *base symbols*. The aim of the calculi is to eliminate the non-base symbols from the input problem. The ordering $>$ specifies in which sequence the

Table 1. The calculus MA^{sw}

Ackermann: $\dfrac{\{\alpha_1 \vee p, \ldots, \alpha_n \vee p\} \cup N(p)}{(N^p_{\sim\alpha_1 \vee \ldots \vee \sim\alpha_n})^{\neg\neg\alpha_1, \ldots, \neg\neg\alpha_n}_{\alpha_1, \ldots, \alpha_n}}$

provided (i) p is a non-base symbol, (ii) p is strictly maximal with respect to each α_i, and (iii) N is negative with respect to p. We refer to the clauses $\alpha_1 \vee p$, $\ldots$, $\alpha_n \vee p$ as the *positive premises* of the rule.

Surfacing: $\dfrac{N \cup \{\alpha \vee \Box^\sigma \beta(p)\}}{N \cup \{\Box^{\sigma,\smile} \alpha \vee \beta(p)\}}$

provided (i) p is the largest non-base symbol which occurs in $\alpha \vee \Box^\sigma \beta$, (ii) p does not occur in α, and (iii) $\Box^\sigma \beta$ is positive with respect to p. It is assumed that σ is non-empty.

Skolemization: $\dfrac{N \cup \{\neg a \vee \neg\Box^\sigma \beta(p)\}}{N \cup \{\neg a \vee \neg\Box^\sigma \neg b, \neg b \vee \sim\beta(p)\}}$

provided (i) p is the largest non-base symbol which occurs in $\neg a \vee \neg\Box^\sigma \beta$, (ii) $\neg\Box^\sigma \beta$ is positive with respect to p and (iii) b is a new nominal. It is assumed that σ is non-empty.

Clausify: $\dfrac{N \cup \{\neg(\alpha \vee \beta)\}}{N \cup \{\sim\alpha, \sim\beta\}}$

Purify: $\dfrac{N(p)}{N^p_\top}$ $\qquad$ $\dfrac{N(p)}{(N^p_{\neg\top})^{\neg\top}_\top}$

provided p is a non-base symbol, N is positive with respect to p, for the left rule, and N is negative with respect to p, for the right rule.

Sign switching: $\dfrac{N(p)}{(N^p_{\neg p})^{\neg p}_p}$

provided (i) N is closed with respect to the other rules, (ii) p is the maximal non-base symbol in N, and (iii) sign switching with respect to p has not been performed before.

non-base symbols are eliminated. We assume $>$ is defined by: $p_i > p_j$ if $1 \leq i < j \leq m$. The elimination operations eliminate strictly maximal symbols first. We say p is *strictly maximal* with respect to a formula α if for any propositional symbol q in α, $p > q$. $\succ$ is any reduction ordering, i.e., a well-founded, rewrite relation, on formulae and clauses which is compatible with the ordering $>$ of the non-base symbols.

Let MA be the calculus comprising the rules in Table 1, except for the sign switching rule. The calculus with the sign switching rule is denoted by MA^{sw}. We use the subscript $_{red}$ for the calculi with the redundancy elimination rules of Table 2. Brackets in $^{(sw)}$ and $_{(red)}$ indicate optionality.

In the MA calculus the Ackermann rule is one of two rules which eliminate non-base symbols. Condition (ii) of the rule implies that p does not occur in $\alpha_1, \ldots, \alpha_n$ and no non-base symbol occurring in any of the α_i is larger than p. By repeated use of the surfacing rule, positive occurrences of maximal non-base symbols are moved upward in the formula tree, so that this formula can be used as positive premise of the Ackermann rule. Note that if α is empty then this is interpreted as $\neg\top$ and $\Box^\sigma \beta$ is replaced by $\Box^{\sigma,\smile} \neg\top \vee \beta$. The Skolemization rule expands the existential formula in a clause of the form $\neg a \vee \neg\Box^\sigma \alpha$. Since a

Table 2. Redundancy elimination rules

Delete:	$\frac{N \cup \{\alpha\}}{N}$	provided α is redundant with respect to N.
Replace:	$\frac{N \cup \{\alpha\}}{N \cup \{\beta\}}$	provided (i) $M \models \alpha$ iff $M \models \beta$, for any $\mathbf{K}^n_{(m)}(\smile)$-model M, and (ii) $\alpha \succ \beta$.

denotes a nominal this formula is interpreted as $M, v(a) \models \neg\Box^\sigma\alpha$ and can be viewed as a *locally satisfiable existential* formula. The clausify rule replaces any top-level conjunction by the set of conjuncts appearing in the conjunctions. The purify rule is the other elimination rule. It eliminates a non-base symbol that occurs only positively or only negatively by substitution with $\top$ or $\neg\top$. The sign switching rule is only applied when it has not been possible to eliminate the maximal non-base symbol in the set. The strategy implicit in the definition is to postpone the application of the rule as much as possible. This is however not essential for correctness or termination of the calculus.

The delete rule (Table 2) is based on the following notion of redundancy. By definition, a formula α is *redundant* with respect to a set N of clauses iff there is a finite subset $\{\beta_1, \ldots, \beta_l\}$ of N such that (i) for any $\mathbf{K}^n_{(m)}(\smile)$-model M, if $M \models \beta_1 \wedge \ldots \wedge \beta_l$ then $M \models \alpha$, and (ii) $\alpha \succ \beta_i$ for each $1 \leq i \leq l$. It is possible to define $\succ$ in such a way that standard simplifications are possible. The replace rule allows the replacement of formulae with logically equivalent formulae. The replaced formulae are in fact redundant. Testing redundancy is in general a computationally hard problem, in fact it has at least the complexity of the base logic. For $\mathbf{K}^n_{(m)}(\smile)$ the complexity is EXPTIME-complete. The delete and replace rules have the advantage that they provide flexibility for using suitable, tractable instances of the rules. We apply redundancy elimination only when β is smaller than α with respect to the ordering $\succ$. Intuitively, β is then "simpler" than α with respect to the ordering. Table 3 lists examples of logical equivalences in $\mathbf{K}^n_{(m)}(\smile)$ that when used as rewrite rules from left-to-right replace formulae by

Table 3. Sample rewrite rules ($\sigma \neq \epsilon$)

Eliminate $\leftrightarrow$, $\rightarrow$, $\wedge$, $\Diamond^\sigma$, $\bot$:		
$\alpha \leftrightarrow \beta \Rightarrow \neg(\neg(\sim\alpha \vee \beta) \vee \neg(\sim\beta \vee \alpha))$		$\alpha \rightarrow \beta \Rightarrow \sim\alpha \vee \beta$
$\neg(\alpha \leftrightarrow \beta) \Rightarrow \neg(\sim\alpha \vee \beta) \vee \neg(\sim\beta \vee \alpha)$		$\neg(\alpha \rightarrow \beta) \Rightarrow \sim\alpha \vee \sim\beta$
$\alpha \wedge \beta \Rightarrow \neg(\sim\alpha \vee \sim\beta)$	$\Diamond^\sigma\alpha \Rightarrow \neg\Box^\sigma\sim\alpha$	$\bot \Rightarrow \neg\top$
$\neg(\alpha \wedge \beta) \Rightarrow \sim\alpha \vee \sim\beta$	$\neg\Diamond^\sigma\alpha \Rightarrow \Box^\sigma\sim\alpha$	$\neg\bot \Rightarrow \top$
Distributivity of $\vee$, $\Box^\sigma$ over $\wedge$:		Obvious simplifications:
$\neg(\alpha \vee \beta) \vee \gamma \Rightarrow \neg(\neg(\sim\alpha \vee \gamma) \vee \neg(\sim\beta \vee \gamma))$		$\alpha \vee \alpha \Rightarrow \alpha$
$\neg(\neg\Box^\sigma\alpha \vee \neg\Box^\sigma\beta) \Rightarrow \Box^\sigma\neg(\sim\alpha \vee \sim\beta)$		$\alpha \vee \neg\alpha \Rightarrow \top$
Simplifications involving $\Box^\sigma$ and $\Box^{\sigma,\smile}$:		$\alpha \vee \top \Rightarrow \top$
$\neg a \vee \neg\Box^\sigma\neg\Box^{\sigma,\smile}\neg a \vee \beta \Rightarrow \neg a \vee \beta$		$\alpha \vee \neg\top \Rightarrow \alpha$
$\alpha \vee \Box^\sigma\neg\Box^{\sigma,\smile}\alpha \Rightarrow \top$		$\Box^\sigma\top \Rightarrow \top$
$\alpha \vee \Box^\sigma\Box^{\sigma,\smile}\alpha \Rightarrow \alpha \vee \Box^\sigma\neg\top$		$\neg\neg\alpha \Rightarrow \alpha$

equivalent smaller formulae which can be implemented efficiently. Ignoring the distributivity of disjunction over conjunction (which can lead to an exponential blow-up), the transformations can be implemented with constant overhead. In the remainder of the paper we assume that only effectively computable instances of the delete and replace rules are used.

For the next theorem we need the notion of Skolem formulae. Suppose b is the nominal introduced by the Skolemization rule reducing $\neg a \vee \neg\Box^{\sigma}\alpha$ to $\neg a \vee \neg\Box^{\sigma}\neg b$ and $\neg b \vee \neg\alpha$. Then the formula $(\neg a \vee \neg\Box^{\sigma}\alpha) \rightarrow ((\neg a \vee \neg\Box^{\sigma}\neg b) \wedge (\neg b \vee \neg\alpha))$ is the *Skolem formula* for b.

Subsequently we always assume N is the set of input clauses and $\Sigma = \{p_1, \ldots, p_m\}$ is the ordered set of non-base symbols we want to eliminate. We say a derivation in $\mathsf{MA}^{(sw)}_{(red)}$ is *successful* if none of the non-base symbols occur in the result N_∞ of the derivation, and a derivation is *unsuccessful*, otherwise.

Theorem 1 (Correctness and termination of $\mathsf{MA}^{(sw)}_{(red)}$). *For any $\mathsf{MA}^{(sw)}_{(red)}$-derivation $N_0(= N), N_1, N_2, \ldots$ from N with result N_∞: (i) No rules are applicable to N_∞ with respect to Σ; (ii) There is an $n \geq 0$ such that $N_n = N_\infty$; (iii) If the derivation is successful, then for any Σ-equivalent models M and M', $M \models \bigwedge N \wedge \bigwedge S(a_1, \ldots, a_l)$ iff $M' \models \bigwedge N_\infty(a_1, \ldots, a_l)$, where $a_1 \ldots a_l$ are the nominals introduced during the derivation and $S(a_1, \ldots, a_l)$ is the set of Skolem formulae for $a_1, \ldots, a_l$.*

If (iii) holds we say that N_∞ *corresponds to* $\exists\Sigma[N]$ *(modulo Skolem formulae).*

The theorem says that any $\mathsf{MA}^{(sw)}$-derivation with respect to symbols in Σ terminates, and when the non-base symbols have been successfully eliminated then the input set is equivalent to the resulting set in the sense of (iii). (iii) is a consequence of the property that all rules preserve equivalence modulo second-order quantification of non-base symbols. For the Ackermann rule the preservation of this equivalence follows from a specialization of Ackermann's Lemma [1] for second-order logic to modal logic. Namely:

Theorem 2 (Ackermann Lemma). *Let α and β be $\boldsymbol{K}^n_{(m)}(\smile)$-formulae and suppose the propositional symbol p does not occur in α. Let M be an arbitrary $\boldsymbol{K}^n_{(m)}(\smile)$-model. If p occurs only negatively in β then $M \models \beta^p_\alpha$ iff there is a model M' which is p-equivalent to M and $M' \models (\alpha \rightarrow p) \wedge \beta(p)$.*

Theorem 1 says in fact that for any given set N of formulae every MA-derivation (with or without the sign switching rule and with or without the redundancy elimination rules) stops after finitely many steps, i.e., termination is always guaranteed. It is however not guaranteed that there is a sequence of transformations that succeeds for the particular ordering of non-base symbols. As a consequence, in general, it may be necessary to attempt all possible orderings of the non-base symbols. Even when all possible orderings are tried, success cannot be guaranteed because there is no rule for bringing non-base symbols occurring below sequences of modal operators to the surface where a diamond operator occurs below a box operator.

4 Quantifier Elimination Procedures

Next we turn the MA^{sw}_{red}-calculus into a procedure, called MSQEL (instead of MA^{sw}_{red} we can also use one of the other calculi). Suppose that α is a given $\mathbf{K}^n_{(m)}(\smile)$-formula. The aim is to eliminate the non-base symbols in Σ from α. The MSQEL procedure involves two stages:

1. *Pre-process input*: While performing simplifications (e.g., based on the rules of Table 3, except for the distributivity rules), transform the input formula α into modal disjunctive normal form. That is, α is transformed into a disjunction of formulae of the form $\beta \wedge \bigwedge_j \Box^{\kappa_j}\gamma_j \wedge \bigwedge_l \neg\Box^{\kappa_l}\delta_l$, where β is a conjunction of propositional literals, and both γ_j and δ_l are in modal disjunctive normal form. If one of the top-level disjuncts is a negated nominal $\neg a$, then pick one of these, say $\neg a$, delete it but add it to the other top-level disjuncts. E.g., $\alpha = \neg a \vee \neg b \vee \alpha'_1 \vee \alpha'_2$ becomes $(\neg a \vee \neg b) \vee (\neg a \vee \alpha_1) \vee (\neg a \vee \alpha_2)$. Suppose the result is the formula $\bigvee_i \alpha_i$.
2. *Reduce disjuncts*: This stage takes each disjunct α_i in turn, selects an ordering of the non-base symbols in Σ and applies the rules of the calculus MA^{sw}_{red} to the set $\{\alpha_i\}$ with respect to this ordering. If this succeeds and returns a set N_i of $\mathbf{K}^n_{(m)}(\smile)$-clauses (which are free of non-base symbols) then we say that MSQEL has *successfully reduced* α_i to N_i. If this stage is unsuccessful then construct a derivation for α_i with respect to a different ordering of the non-base symbols.

The motivation for the pre-processing stage is to improve the success rate of the elimination process, because it allows smaller formulae to be processed in the separate reductions by MA^{sw}_{red}. Whenever the simplifications performed are effective, the pre-processing stage can be performed effectively for any $\mathbf{K}^n_{(m)}(\smile)$-formula. The worst-case complexity of this stage is in general bounded by at least an exponential function in the size of the formula. Observe however that for the preservation of logical equivalence of the entire procedure, pre-processing is not essential. Nevertheless the transformation is useful because it means that smaller formulae are considered in the reduction stage. The transformation to disjunctive normal form is in fact essential for the termination results of Sahlqvist formulae and monadic inductive formulae. The purpose of the 'distribution' of a negated nominal $\neg a$ is to maximize the number of clauses of the form $\neg a \vee \alpha'$ passed to the reduction stage, because more rules can be invoked.

Theorem 3 (Correctness and termination of Msqel). *For any $\boldsymbol{K}^n_{(m)}(\smile)$-formula α and $\Sigma \subseteq V$: (i) Any implementation of* MSQEL *terminates; (ii) If it terminates successfully and returns a family of sets $\langle N_i\rangle_i$ then: (a) $\langle N_i\rangle_i$ is a bounded family of bounded sets of $\boldsymbol{K}^n_{(m)}(\smile)$-formulae free of symbols in Σ; (b) For any model M there is a Σ-equivalent model M' such that $M \models \alpha \wedge \bigwedge S$ iff $M' \models \bigvee_i \langle \bigwedge N_i\rangle_i$, where S is the set of modal Skolem formulae for nominals introduced during the execution of the procedure.*

If (ii.b) holds we say that $\bigvee_i \langle \bigwedge N_i\rangle_i$ *corresponds to* $\exists\Sigma[\alpha]$ *(modulo Skolem formulae).*

For computing first-order correspondence properties the formulae returned by MSQEL need to be translated to first-order logic. Let MSQEL$^\pi$ be the procedure which consists of the pre-processing and reduction stages of MSQEL plus the following translation stage.

3. *Translate to first-order logic*: This stage is performed only when every disjunct α_i has been successfully reduced to a set N_i of $\mathbf{K}^n_{(m)}(\smile)$-formulae. Each set is first transformed into a set M_i of first-order formulae in the obvious way using the standard translation π. It remains to eliminate the constants corresponding to the nominals introduced by applications of the surfacing rule. This can be done by unskolemization which is always successful. Hence, the procedure terminates successfully and returns the formula $\forall x\big[\bigvee_i \mathrm{UnSk}(M_i)\big]$, where UnSk denotes the unskolemization operator and $\forall x$ binds the free variable that may occur in each M_i.

Theorem 4 (Correctness and termination of Msqel$^\pi$). *For any $\boldsymbol{K}^n_{(m)}(\smile)$-formula α and $\Sigma \subseteq V$: (i) Any implementation of* MSQEL$^\pi$ *terminates; (ii) If* MSQEL$^\pi$ *terminates successfully and returns the formula β then: (a) β is a first-order formula; (b) For any model M there is a Σ-equivalent model M' such that $M \models \alpha$ iff $M'^* \models \beta$, where M'^* is the first-order model which corresponds to M'.*

When condition (ii.b) holds, we say that β *corresponds to* $\exists\Sigma[\alpha]$. In this case we also say $\neg\beta$ *corresponds to* $\forall\Sigma[\neg\alpha]$.

5 Computing Correspondences

MSQEL$^\pi$ can be used to compute first-order correspondence properties for modal axioms and modal rules. This is how:

1. Take the given rule Δ/Δ' and pass the formula $\beta = \bigwedge\Delta \wedge \bigwedge\{\neg a \vee \neg\alpha \mid \alpha \in \Delta'\}$, where a is a fresh nominal, to MSQEL$^\pi$. a is the Skolem constant associated with the quantifier in the negation of the succedent of $\Pi(\Delta/\Delta')$. The aim is to eliminate all propositional symbols that occur in β, i.e., $\Sigma = V$.
2. If MSQEL$^\pi$ succeeds, suppose it returns the formula γ.

By Theorem 4 we have that: $\neg\gamma$ corresponds to the rule Δ/Δ'. (In fact, if $\Delta = \emptyset$ then $\neg\gamma$ is a local frame correspondence property for the axiom $\bigvee\Delta'$.) With MSQEL instead of MSQEL$^\pi$ the derived formula is a pure $\mathbf{K}^n_{(m)}(\smile)$-formula.

For illustration we consider two examples; more examples can be found in the long version. First, the first-order correspondence property of $\forall p\forall q[\Box(\Box p \leftrightarrow q) \rightarrow \Diamond\Box\neg p]$ is $\forall x\exists y\forall z[R(x,y) \wedge \neg R(y,z)]$. Negating the modal formula gives:

~~1.~~ $\neg a \vee \Box(\Box p \leftrightarrow q)$

2. $\neg a \vee \neg\Diamond\Box\neg p$

We want to eliminate the variables p and q. Using the ordering $p > q$ the reduction by MA^{sw}_{red} does not succeed (cf. long version). MSQEL$^\pi$ now tries to reduce the problem using a different ordering, namely $q > p$.

~~3.~~ $\Box^{\smile}\neg a \vee (\Box p \leftrightarrow q)$	1, surf.
~~4.~~ $\Box^{\smile}\neg a \vee \neg\Box p \vee q$	3, repl., cl.
~~5.~~ $\Box^{\smile}\neg a \vee \Box p \vee \neg q$	3, repl., cl.
~~6.~~ $\Box^{\smile}\neg a \vee \Box p \vee \Box^{\smile}\neg a \vee \neg\Box p$	4 into 5, Acker.
7. $\top$	6, repl.
8. $\neg a \vee \neg\Diamond\Box\neg\top$	$\{2\}$, purify

The procedure returns the first-order translation of the negation of $\neg a \vee \neg\Diamond\Box\neg\top$ which gives the property we expect to obtain.

The example shows the general sensitivity to the order in which variables are eliminated and the importance of detecting redundancy, without which the second attempt would fail, too. The example is not a Sahlqvist formula or a monadic inductive formula, but can, as we see, still be solved.

The second example illustrates how modal rules can be reduced. The rule $\Box\Box p/\Diamond p$ corresponds to $\forall x\exists y\exists z[R(x,y) \wedge R^2(z,y)]$. A derivation is:

~~1.~~ $\Box^2 p$	
~~2.~~ $\neg a \vee \neg\Diamond p$	
~~3.~~ $\Box^{2,\smile}\neg\top \vee p$	1, surf.
~~4.~~ $\neg a \vee \Box\neg p$	2, repl.
5. $\neg a \vee \Box\Box^{2,\smile}\neg\top$	3 into 4, Acker.

Now translate clause 5 into first-order logic and negate.

6 The Classes $\mathcal{C}$ and $\mathcal{C}^{>}$

Next we define the classes $\mathcal{C}$ and $\mathcal{C}^{>}$ for which we can guarantee *successful* termination.

Assume $\Sigma = \{p_1, \ldots, p_m\} \subseteq V$ is the set of propositional non-base symbols. Let p be a propositional variable (not necessarily in Σ). A formula in the following form

$$\Box^{\sigma_1}(\beta_1 \vee \Box^{\sigma_2}(\beta_2 \vee \ldots \Box^{\sigma_n}(\beta_n \vee p)\ldots))$$

is called a *universal formula (positive) in* p, if $\beta_1, \ldots, \beta_n$ are negative formulae in Σ and each $\Box^{\sigma_i}$ is a possibly empty sequence of box operators ($1 \leq i \leq n$). Let N be a set of universal formulae. The *dependency relation* $\succ_d$ over occurrences of the non-base symbols is defined by: $p \succ_d q$ iff there is $\gamma \in N$, p is a positive occurrence in γ and q is a negative occurrence in γ. Let $\succ_d^+$ be the transitive closure of $\succ_d$. If there is no variable occurrence of p such that $p \succ_d^+ p$ then we say that there are *no cycles over* Σ in N. (Note, the dependency relation $\succ_d$ should not be confused with $\succ$ or the ordering $>$ on Σ.)

We define the class $\mathcal{C}$ as follows. Let N be a set of $\mathbf{K}^n_{(m)}(\smile)$-clauses and let Σ be a non-empty set of propositional symbols. A pair $\langle N, \Sigma\rangle$ belongs to $\mathcal{C}$ if the following conditions hold:

1. Each clause in N is a clause of one of these forms:

   negative clause: β; *universal clause*: γ; *local clause*: $\neg b \vee \neg\Box^{\sigma}\delta$;

   where $\sigma \neq \epsilon$, b denotes a nominal, β a negative clause[1] in Σ, γ a universal clause, and δ either a negated universal formula or a disjunction of negated universal formulae and positive formulae in Σ.
2. There are no cycles over Σ in N.

Theorem 5. *For any $\langle N, \Sigma\rangle \in \mathcal{C}$: (i) For any ordering of the variables in Σ, any derivation based on* MA *effectively reduces N to a set N' of $\boldsymbol{K}^{n}_{(m)}(^{\smile})$-formulae which are free of symbols in Σ, and N' corresponds to $\exists\Sigma[N]$ (modulo Skolem formulae); (ii) Any* MSQEL$^{\pi}$ *derivation effectively computes a first-order formula which corresponds to $\exists\Sigma[N]$.*

The theorem says that the class $\mathcal{C}$ is equivalently reducible to formulae defined only over the base symbols. Moreover, these can be computed with the rules of MA, i.e., without sign switching, redundancy and using any ordering. This means that MSQEL and MSQEL$^{\pi}$ are successful without needing to attempt different orderings. Note however that for efficiency reasons the ordering of the non-base symbols and the order of rule application do matter.

Since the ordering of the non-base symbols, sign switching and redundancy is irrelevant for the success of the elimination of the non-base symbols, it is possible to define an even larger class of formulae for which these *are* relevant. $\mathcal{C}^{>}$ is such a class.

We define $\mathcal{C}^{>}$ as a class of tuples $\langle N, \Sigma, >\rangle$, where N is a set of $\mathbf{K}^{n}_{(m)}(^{\smile})$-clauses, $\Sigma = \{p_1, \ldots, p_m\}$ is an ordered set of non-base symbols and $>$ is the ordering of Σ. By definition, $\langle N, \Sigma, >\rangle \in \mathcal{C}^{>}$ iff there is a sequence $N_1(= N), N_2, \ldots, N_m$ of sets of clauses, such that

1. $(N_i, \{p_i\})$ each belong to $\mathcal{C}$, and
2. each N_{i+1} is the output of MA^{sw}_{red} on the input N_i and $\{p_i\}$, where $1 \leq i \leq m$.

Consider the set of clauses $N = \{2, 4, 5\}$ to which the first example in Section 5 can be reduced. Let $\Sigma = \{q, p\}$ and $q > p$. While $\langle N, \Sigma\rangle$ does not belong to $\mathcal{C}$, $\langle N, \Sigma, >\rangle$ does belong to $\mathcal{C}^{>}$. This shows that $\mathcal{C}^{>}$ strictly subsumes $\mathcal{C}$.

Theorem 6. *For any $\langle N, \Sigma, >\rangle \in \mathcal{C}^{>}$: (i) Using the ordering $>$ of the non-base symbols in Σ, MA^{sw}_{red} effectively reduces N to a set N' of $\boldsymbol{K}^{n}_{(m)}(^{\smile})$-formulae which are free of symbols in Σ and N' corresponds to $\exists\Sigma[N]$ (modulo Skolem formulae); (ii)* MSQEL$^{\pi}$ *effectively computes a first-order formula which corresponds to $\exists\Sigma[N]$.*

How do the Sahlqvist's class [9] and the more general class of monadic inductive axioms [3] relate to $\mathcal{C}$ and $\mathcal{C}^{>}$? The next result tells us that $\mathcal{C}$ accommodates both the Sahlqvist class and the class of monadic inductive formulae. For both the latter $\Sigma = V$. Thus $\mathcal{C}$ represents a strictly larger set of problems.

[1] A clause with a property, e.g. being negative, universal, etc, is a clause which is a formula with that property.

Lemma 1. *Any negated Sahlqvist formula over* $\boldsymbol{K}^n_{(m)}(\smile)$ *and any negated monadic inductive formula over* $\boldsymbol{K}^n_{(m)}(\smile)$ *can be reduced by standard equivalence preserving transformations to a set of clauses in* $\mathcal{C}$*.*

Theorem 7. *Let* α *be any Sahlqvist formula over* $\boldsymbol{K}^n_{(m)}(\smile)$*. Then: (i) Any derivation of* MSQEL *on* $\neg a \vee \neg\alpha$ *successfully computes an equivalent, correspondent* $\boldsymbol{K}^n_{(m)}(\smile)$*-formula which is free of any propositional variables; (ii) Any derivation of* MSQEL$^\pi$ *on* $\neg a \vee \neg\alpha$ *successfully computes an equivalent first-order correspondent. In both cases any ordering may be used, and sign switching and redundancy elimination are optional.*

Theorem 8. *The appropriate reformulation of Theorem 7 is true for any monadic inductive formula over* $\boldsymbol{K}^n_{(m)}(\smile)$*.*

Based on our analysis it is possible to define syntactic classes of modal rules which are equivalently reducible to first-order correspondence properties. See the long version for details.

7 Conclusion

The setting of our investigation was modal tense logic but all the techniques and results are based on general principles which apply to polyadic modal logics including polyadic inductive formulae defined in [8]. The ideas and results also carry over to other logics. For example, it is immediate that all the results transfer to description and hybrid logics which correspond to the logics considered in this paper. The integration of the ideas and techniques into methods for first-order logic and classical fixpoint logic (DLS) does not pose any technical difficulties either.

References

1. Ackermann, W.: Untersuchung über das Eliminationsproblem der mathematischen Logik. Math. Ann. 110, 390–413 (1935)
2. Bachmair, L., Ganzinger, H., Waldmann, U.: Refutational theorem proving for hierarchic first-order theories. Appl. Algebra Engineering, Comm. Computing 5(3/4), 193–212 (1994)
3. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic: I. The core algorithm SQEMA. J. Logic Computat. 2(1-5), 1–26 (2006)
4. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic: II. Polyadic and hybrid extensions of the algorithm SQEMA. J. Logic Comput. 16, 579–612 (2006)
5. Doherty, P., Lukaszewicz, W., Szałas, A.: Computing circumscription revisited: A reduction algorithm. J. Automat. Reason. 18(3), 297–336 (1997)
6. Gabbay, D.M., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. S. Afr. Computer J. 7, 35–43 (1992)

7. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publ. (2008)
8. Goranko, V., Vakarelov, D.: Elementary canonical formulae: Extending Sahlqvist's theorem. Ann. Pure Appl. Logic 141(1-2), 180–217 (2006)
9. Sahlqvist, H.: Completeness and correspondence in the first and second order semantics for modal logics. In: Proc. 3rd Scandinavian Logic Symposium, 1973, pp. 110–143. North-Holland, Amsterdam (1975)

Literal Projection for First-Order Logic

Christoph Wernhard

Universität Koblenz-Landau
wernhard@uni-koblenz.de

Abstract. The computation of literal projection generalizes predicate quantifier elimination by permitting, so to speak, quantifying upon an arbitrary sets of ground literals, instead of just (all ground literals with) a given predicate symbol. Literal projection allows, for example, to express predicate quantification upon a predicate just in positive or negative polarity. Occurrences of the predicate in literals with the complementary polarity are then considered as unquantified predicate symbols. We present a formalization of literal projection and related concepts, such as literal forgetting, for first-order logic with a Herbrand semantics, which makes these notions easy to access, since they are expressed there by means of straightforward relationships between sets of literals. With this formalization, we show properties of literal projection which hold for formulas that are free of certain links, pairs of literals with complementary instances, each in a different conjunct of a conjunction, both in the scope of a universal first-order quantifier, or one in a subformula and the other in its context formula. These properties can justify the application of methods that construct formulas without such links to the computation of literal projection. Some tableau methods and direct methods for second-order quantifier elimination can be understood in this way.

1 Introduction

Predicate quantifier elimination has a large variety of applications in knowledge processing, which continue to become apparent since the early nineties until very recently [1,2,3,4,5,6,7,8]. This is sometimes not obvious, because operations such as the computation of uniform interpolants, forgetting and projection are in fact variants of predicate quantifier elimination. In parallel to discovering applications, the development of methods that perform predicate quantifier elimination in first-order and related logics has been of continued interest since the early nineties [1,3,9,8]. In recent years also predicate quantifier elimination in propositional logic became a subject of research, driven largely by advances in SAT-solving [4,5,10,11,12].

In this paper we focus on *literal projection*, which generalizes predicate quantification by permitting, so to speak, quantifying upon an *arbitrary set of ground literals*, instead of just (all ground literals with) a given predicate symbol. Literal projection allows, for example, to express predicate quantification upon a

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 389–402, 2008.

predicate just in positive or negative polarity. Eliminating such a quantifier from a formula in negation normal form results in a formula that might still contain the quantified predicate, but only in literals whose polarity is complementary to the quantified one. The result of wrapping a formula into such an existential quantifier, followed by eliminating it, has – among the formulas that do not contain the quantified predicate in the quantified polarity – exactly the same theorems as the original formula. The result can be considered as an extract of the original formula, where knowledge about the quantified predicate in the quantified polarity is "forgotten", but all other knowledge is retained. A look over the 40000 theorems in the Mizar mathematical library, the largest collection of formalized mathematical knowledge, indicates the order of magnitude of the role of asymmetric polarity in actual knowledge bases: In 98.2 percent of the theorems, at least one predicate symbol occurs only in a single polarity. In 66.9 percent, no predicate symbol occurs in both polarities. On average, 0.89 percent of the predicate symbols in the signature of a theorem occur in the theorem only in a single polarity.[1]

Literal forgetting, a variant of literal projection, has been formalized for propositional logic in [11]. We generalize this formalization to first-order logic with Herbrand interpretations. A new formulation of the characterization in [11] facilitates the formal access to literal projection and related notions, which then can be expressed by means of straightforward relationships between sets of literals. With this formalization, we show some properties of literal projection which hold for formulas that are free of certain *links*, pairs of literals with complementary instances, each in a different conjunct of a conjunction, both in the scope of a universal first-order quantifier, or one in a subformula and the other in its context formula. These properties can justify the application of methods that construct formulas without such links to predicate quantifier elimination, or, more generally, to the computation of literal projection. Some tableau construction procedures and direct methods [8] for second-order quantifier elimination can be understood in this way.

The structure of the paper is as follows: In Sect. 2 the semantic framework with the characterization of literal projection is defined and illustrated by some examples. Further related concepts are defined in Sect. 3, in particular a formal account of the set of literals "about which a formula expresses something." In Sect. 4 basic properties of literal projection are summarized, properties that relate to linklessness and conjunction are developed, and their application to computation methods is outlined. A further property that relates to linklessness between a subformula and its context is discussed in Sect. 5. In the conclusion, applications of literal projection in knowledge-based systems are suggested.

[1] These statistics have been obtained with the MPTP 0.2 translation of the Mizar library [13]. About 1000 theorems which have just equality as predicate or translate to *true* have not been considered. Predicates that are only implicit in the original Mizar syntax (*modes*, *attributes* and *aggregates*), as well as predicate occurrences in set abstractions and type specifiers, have been taken into account.

2 Semantic Framework, Projection and Forgetting

Notation. We write the positive (negative) literal with atom A as $+A$ ($-A$). We understand a *first-order formula* as in negation normal form, constructed from literals, truth value constants $\top, \bot$, binary connectives $\wedge, \vee$, and quantifiers $\forall, \exists$. Implication $\rightarrow$, negation $\neg$ and dropping the sign of positive literals are understood as meta-level notation with respect to this syntax. We assume a fixed first-order signature with at least one constant symbol. The sets of all ground terms and all ground atoms, with respect to this signature, are denoted by GTERMS and GATOMS. Variables are x, y, z, also with subscripts. To avoid clumsy handling of quantifier scopes, we assume that in a formula all occurrences of the same variable are either free or are bound by an occurrence of a quantifier, and that no two quantifier occurrences bind occurrences of the same variable.

The Projection Operator and Literal Scopes. A *formula* in general is like a first-order formula, but in its construction a further operator, $\mathsf{project}(F, S)$, is permitted, where F is a formula and S specifies a set of ground literals. We call a set of ground literals in the role as argument to $\mathsf{project}$ a *literal scope*. The formula $\mathsf{project}(F, S)$ is called the *literal projection* of F onto S. Literal projection generalizes existential second-order quantification. It is further discussed below.

Interpretations. We characterize the semantics with a notational variant of the framework of Herbrand interpretations: An *interpretation* is a pair $\langle I, \beta \rangle$, where I is a *structure*, that is, a set of ground literals that contains for all ground atoms A exactly one of $+A$ or $-A$, and β is a *variable assignment*, that is, a mapping of the set of variables into GTERMS.

Satisfaction Relation. The satisfaction relation between interpretations $\langle I, \beta \rangle$ and formulas is defined by the clauses in Tab. 1, where L matches a literal, F, F_1, F_2 match a formula, and S matches a literal scope specifier. Two operations are defined on variable assignments β : If F is a formula, then $F\beta$ denotes F with all variables replaced by their image in β. If x is a variable and t a ground term, then $\beta\frac{t}{x}$ is the variable assignment that maps x to t and all other variables to the same values as β.

The semantic definition of literal projection in Tab. 1 gives a formal account of the following more intuitive characterization: An interpretation $\langle I, \beta \rangle$ satisfies $\mathsf{project}(F, S)$ if and only if there is a structure J such that $\langle J, \beta \rangle$ satisfies F and I can be obtained from J by replacing literals that are not in S with their complements. This includes the special case $I = J$, where no literals are replaced.

Entailment and equivalence can be straightforwardly defined in terms of the satisfaction relation: A formula F_1 *entails* a formula F_2, in symbols $F_1 \models F_2$, if and only if for all interpretations $\langle I, \beta \rangle$ it holds that if $\langle I, \beta \rangle \models F_1$ then $\langle I, \beta \rangle \models F_2$. A formula F_1 is *equivalent* to a formula F_2, in symbols $F_1 \equiv F_2$, if and only if $F_1 \models F_2$ and $F_2 \models F_1$.

Relation to Conventional Model Theory. Literal sets as components of interpretations permit the straightforward definition of the semantics of

Table 1. The Satisfaction Relation with the Semantic Definition of Literal Projection

$\langle I,\beta\rangle \models \top$		
$\langle I,\beta\rangle \not\models \bot$		
$\langle I,\beta\rangle \models L$	iff_{def}	$L\beta \in I$
$\langle I,\beta\rangle \models F_1 \wedge F_2$	iff_{def}	$\langle I,\beta\rangle \models F_1$ and $\langle I,\beta\rangle \models F_2$
$\langle I,\beta\rangle \models F_1 \vee F_2$	iff_{def}	$\langle I,\beta\rangle \models F_1$ or $\langle I,\beta\rangle \models F_2$
$\langle I,\beta\rangle \models \forall x\, F$	iff_{def}	for all $t \in \mathsf{GTERMS}$ it holds that $\langle I,\beta\frac{t}{x}\rangle \models F$
$\langle I,\beta\rangle \models \exists x\, F$	iff_{def}	there exists a $t \in \mathsf{GTERMS}$ such that $\langle I,\beta\frac{t}{x}\rangle \models F$
$\langle I,\beta\rangle \models \mathsf{project}(F,S)$	iff_{def}	there exists a structure J such that $\langle J,\beta\rangle \models F$ and $J \cap S \subseteq I$

literal projection given in the last clause in Tab. 1. The set of literals I of an interpretation $\langle I,\beta\rangle$ is called *"structure"*, since it can be considered as representation of a structure in the conventional sense used in model theory: The domain is the set of ground terms. Function symbols f with arity $n \geq 0$ are mapped to functions f' such that for all ground terms $t_1,...,t_n$ it holds that $f'(t_1,...,t_n) = f(t_1,...,t_n)$. Predicate symbols p with arity $n \geq 0$ are mapped to $\{\langle t_1,...,t_n\rangle \mid +p(t_1,...,t_n) \in I\}$. Moreover, an interpretation $\langle I,\beta\rangle$ represents a conventional second-order interpretation [14] (if predicate variables are considered as distinguished predicate symbols): The structure in the conventional sense corresponds to I, as described above, except that mappings of predicate variables are omitted. The assignment is β, extended such that all predicate variables p are mapped to $\{\langle t_1,...,t_n\rangle \mid +p(t_1,...,t_n) \in I\}$.

Some More Notation. If L is a literal, S is a literal scope, I is an interpretation, x is a variable, and t is a term, then: $\widetilde{L}$ denotes the *complement* of L; $\widetilde{S} \stackrel{\text{def}}{=} \{\widetilde{L} \mid L \in S\}$; $\overline{S} \stackrel{\text{def}}{=} \mathsf{GLITS} - S$; S is called *consistent* if it does not contain a literal and its complement; $I[L] \stackrel{\text{def}}{=} (I - \{\widetilde{L}\}) \cup \{L\}$; $I[S] \stackrel{\text{def}}{=} (I - \widetilde{S}) \cup S$; $F\{x \mapsto t\}$ is F with all occurrences of x replaced by t.

Literal Forgetting. In some applications it is natural to consider projection onto all literals *with exception* of those in a given set. The concept of *forgetting* allows to express this conveniently: *The literal forgetting in F about S*, in symbols $\mathsf{forget}(F,S)$, is defined by $\mathsf{forget}(F,S) \stackrel{\text{def}}{=} \mathsf{project}(F,\overline{S})$.

Atom Projection and Forgetting. In the special case where the literal scope S is equal to $\widetilde{S}$, we speak of *atom projection* and *atom forgetting*. The condition $J \cap S \subseteq I$ in the semantic definition of $\mathsf{project}$ is then equivalent to $I \cap S = J \cap S$. Existential second-order quantification can be expressed in terms of atom forgetting: $\exists p\, F$ corresponds to $\mathsf{forget}(F, \{L \mid L \textit{ is a ground literal with predicate } p\})$.

By the way, at least for propositional logic, it is also possible to define literal forgetting in terms of atom forgetting, since for propositional formulas F and ground literals L it holds that $\mathsf{forget}(F,\{L\}) \equiv (\mathsf{forget}(L \wedge F, \{L,\widetilde{L}\}) \vee (\widetilde{L} \wedge F))$.

Table 2. Examples of Literal Projection

(i) $M_1 \stackrel{\text{def}}{=} \{+p, +q, +r\}$,
$M_2 \stackrel{\text{def}}{=} \{-p, +q, +r\}$,
$M_3 \stackrel{\text{def}}{=} \{-p, -q, +r\}$,
$M_4 \stackrel{\text{def}}{=} \{-p, -q, -r\}$.

(ii) $M_1 \cap S = \{+p, +q, +r\}$,
$M_2 \cap S = \{-p, +q, +r\}$,
$M_3 \cap S = \{-p, +r\}$,
$M_4 \cap S = \{-p, -r\}$.

(iii) $M'_1 \stackrel{\text{def}}{=} \{+p, +q, +r\}$,
$M'_2 \stackrel{\text{def}}{=} \{-p\}$.

(iv) $M_1 \cap S = \{+p, +r\}$,
$M_2 \cap S = \{-p, +r\}$,
$M_3 \cap S = \{-p, -q, +r\}$,
$M_4 \cap S = \{-p, -q, -r\}$.

(v) $M_1 \cap S = \{+p, +r\}$,
$M_2 \cap S = \{-p, +r\}$,
$M_3 \cap S = \{-p, +r\}$,
$M_4 \cap S = \{-p, -r\}$.

Example 1 (Forgetting a Negative Literal). Let $F \stackrel{\text{def}}{=} ((p \to q) \wedge (q \to r))$ and $S \stackrel{\text{def}}{=} \{+p, -p, +q, +r, -r\}$. We now illustrate that $\mathsf{project}(F, S) \equiv ((p \to q) \wedge (p \to r))$. It is not hard to see that the models of F are exactly those interpretations whose structures are a superset of at least one of $M_1, ..., M_4$, defined as shown in Tab. 2 (i). Thus the equations in Tab. 2 (ii) hold. By the semantic definition of literal projection, $\mathsf{project}(F, S)$ is a formula whose models are exactly the interpretations which are a superset of at least one of the $M_i \cap S$, for $i \in \{1, ..., 4\}$. It is easy to see that this condition on interpretations, being a superset of at least one of the $M_i \cap S$, is equivalent to being a superset of M'_1 or M'_2, defined as in Tab. 2 (iii). It is not hard to see that the models of $((p \to q) \wedge (p \to r))$ are exactly the interpretations that satisfy this condition.

Example 2 (Forgetting a Positive Literal). Let F be defined as in Examp. 1 and $S \stackrel{\text{def}}{=} \{+p, -p, -q, +r, -r\}$. Thus, S is as in Examp. 1, except that it contains q negatively instead of positively. Analogously to Examp. 1, it can be shown that $\mathsf{project}(F, S) \equiv ((p \to r) \wedge (q \to r))$, where, if $M_1, ..., M_4$ are defined as in Examp. 1, the equations in Tab. 2 (iv) hold.

Example 3 (Forgetting an Atom). Let F be defined as in Examp. 1 and $S \stackrel{\text{def}}{=} \{+p, -p, +r, -r\}$. Thus, S is as in Examp. 1 and 2, except that it does not contain a literal with atom q. Analogously to Examp. 1, it can be shown that $\mathsf{project}(F, S) \equiv (p \to r)$, where, if $M_1, ..., M_4$ are defined as in Examp. 1, the equations in Tab. 2 (v) hold.

3 Essential Literal Base and Related Concepts

Literal Base and Essential Literal Base. The signature, that is, the set of predicate and function symbols, of a knowledge base hints the objects, concepts and relations about which it expresses "knowledge". But the signature might be too large: For example the formula $KB = (p \vee (q \wedge \neg q))$ is clearly equivalent to p. Thus KB does express something about p but not about q, although q is in its signature. One might argue that in practice such redundancies might be avoided by carefully engineering knowledge bases. But such redundancies may also arise by combining knowledge bases that are free of them. For example

conjoining $(q \to p)$ with $(\neg q \to p)$ results in an equivalent to KB, which, as we have seen, does not express anything about q, although each of the conjuncts expresses something about q. We call the set of ground literals "about which a formula expresses something" its *essential literal base*, made precise in Def. 2. The essential literal base of KB, for example, is $\{p\}$.

Definition 1 (Literal Base). The *literal base* of a formula F, in symbols $\mathcal{L}(F)$, is the set of ground instances of literals in F.

Definition 2 (Essential Literal Base). The *essential literal base* of a formula F, in symbols $\mathcal{L}_{\mathcal{E}}(F)$, is defined as $\mathcal{L}_{\mathcal{E}}(F) \stackrel{\text{def}}{=} \{L \mid L \text{ is a ground literal and there exist an interpretation } \langle I, \beta\rangle \text{ such that } \langle I, \beta\rangle \models F \text{ and } \langle I[\widetilde{L}], \beta\rangle \not\models F\}$.

The essential literal base of a formula is a subset of its literal base. The essential literal base is independent of syntactic properties: equivalent formulas have the same essential literal base.[2]

Switching Values Outside the [Essential] Literal Base. Prop. 1 below states a property of literal bases that is useful to prove further properties: From a given model of a formula another model can be obtained by switching only literals not in the literal base of the formula. The precondition $S \cap \mathcal{L}(F) = \emptyset$ is equivalent to $\widetilde{S} \cap \widetilde{\mathcal{L}(F)} = \emptyset$. This suggests a second way to read the proposition: another model can be obtained by switching literals in a way such that none of the new values is complementary to an element of the literal base.

Proposition 1. *If $\langle I, \beta\rangle$ is an interpretation, F is a formula and S is a consistent set of ground literals such that $\langle I, \beta\rangle \models F$ and $S \cap \mathcal{L}(F) = \emptyset$, then $\langle I[\widetilde{S}], \beta\rangle \models F$.*

The analog to Prop. 1 for the *essential* literal base can be shown for first-order formulas which do not contain existential quantifiers. For such formulas F, interpretations $\langle I, \beta\rangle$, and consistent sets of ground literals S, it can be proven that if $\langle I, \beta\rangle \models F$ and $\langle I[S], \beta\rangle \not\models F$, then there exists a *finite* set $S' \subseteq S$ such that $\langle I[S'], \beta\rangle \not\models F$. From this property, the analog to Prop. 1 for the essential literal base can be derived. Since this analog is useful to prove properties of projection, we give formulas that satisfy it a name, $\mathcal{E}$-formulas:

Definition 3 ($\mathcal{E}$-Formula). A formula F is called *$\mathcal{E}$-formula* if and only if for all interpretations $\langle I, \beta\rangle$ and consistent sets of ground literals S such that $\langle I, \beta\rangle \models F$ and $S \cap \mathcal{L}_{\mathcal{E}}(F) = \emptyset$ it holds that $\langle I[\widetilde{S}], \beta\rangle \models F$.

As indicated above, first-order formulas without existential quantifier – including propositional formulas and first-order clausal formulas – are $\mathcal{E}$-formulas. Being an $\mathcal{E}$-formula is a property that just depends on the semantics of a formula, that is, an equivalent to an $\mathcal{E}$-formula is also an $\mathcal{E}$-formula.

[2] For propositional formulas, *literal base* and *essential literal base* are defined in [11], called the sets of literals on which a formula is *syntactically* (*semantically*, resp.) *Lit-dependent*.

4 Properties of Projection

Basic Properties. Based on the semantic definition of project, it is not hard to prove properties of projection as displayed in Tab. 3 and 4. They hold for all formulas F, F_1, F_2, $\mathcal{E}$-formulas E, literals L, and literal scopes S, S_1, S_2. The properties in Tab. 4 strengthen properties in Tab. 3, but apply only to $\mathcal{E}$-formulas.

Projection is "semantically determined", in the sense that projections of equivalent formulas onto the same scope are equivalent (Tab. 3.iii). The projection of a formula onto its literal base is equivalent to the original formula (Tab. 3.vii). The essential literal base of a projection is a subset of the projection scope and also a subset of the essential literal base of the projection formula (Tab. 3.xii,xiii). Only the intersection of the given scope with the literal base of the argument formula is relevant for projection and forgetting (Tab. 3.xv,xvi). Property Tab. 3.xvii is behind many applications of predicate quantifier elimination and its variants: A formula *Query* is entailed by a formula *KB* if and only if it is entailed by the projection of *KB* onto the literal base of *Query*. Properties Tab. 3.xviii–xxiii show relationships of projection with other logic operators.

Conjunction and Linklessness. While projection distributes straightforwardly over disjunction and existential first-order quantification (Tab. 3.xx, xxii), only one direction of the corresponding equivalences holds for conjunction and

Table 3. Properties of Projection

(i)	$F \models \mathsf{project}(F, S)$
(ii)	*if* $F_1 \models F_2$, *then* $\mathsf{project}(F_1, S) \models \mathsf{project}(F_2, S)$
(iii)	*if* $F_1 \equiv F_2$, *then* $\mathsf{project}(F_1, S) \equiv \mathsf{project}(F_2, S)$
(iv)	*if* $S_1 \supseteq S_2$, *then* $\mathsf{project}(F, S_1) \models \mathsf{project}(F, S_2)$
(v)	$\mathsf{project}(\mathsf{project}(F, S_1), S_2) \equiv \mathsf{project}(F, S_1 \cap S_2)$
(vi)	$F_1 \models \mathsf{project}(F_2, S)$ *if and only if* $\mathsf{project}(F_1, S) \models \mathsf{project}(F_2, S)$
(vii)	$\mathsf{project}(F, \mathcal{L}(F)) \equiv F$
(viii)	$\mathsf{project}(F, \mathsf{GLITS}) \equiv F$
(ix)	$\mathsf{project}(\top, S) \equiv \top$
(x)	$\mathsf{project}(\bot, S) \equiv \bot$
(xi)	F *is satisfiable if and only if* $\mathsf{project}(F, S)$ *is satisfiable*
(xii)	$\mathcal{L}_\mathcal{E}(\mathsf{project}(F, S)) \subseteq S$
(xiii)	$\mathcal{L}_\mathcal{E}(\mathsf{project}(F, S)) \subseteq \mathcal{L}_\mathcal{E}(F)$
(xiv)	*if* $\mathsf{project}(F, S) \models F$, *then* $\mathcal{L}_\mathcal{E}(F) \subseteq S$
(xv)	$\mathsf{project}(F, S) \equiv \mathsf{project}(F, \mathcal{L}(F) \cap S)$
(xvi)	$\mathsf{forget}(F, S) \equiv \mathsf{forget}(F, \mathcal{L}(F) \cap S)$
(xvii)	$F_1 \models F_2$ *if and only if* $\mathsf{project}(F_1, \mathcal{L}(F_2)) \models F_2$
(xviii)	*if no instance of* L *is in* S, *then* $\mathsf{project}(L, S) \equiv \top$
(xix)	*if all instances of* L *are in* S, *then* $\mathsf{project}(L, S) \equiv L$
(xx)	$\mathsf{project}(F_1 \vee F_2, S) \equiv \mathsf{project}(F_1, S) \vee \mathsf{project}(F_2, S)$
(xxi)	$\mathsf{project}(F_1 \wedge F_2, S) \models \mathsf{project}(F_1, S) \wedge \mathsf{project}(F_2, S)$
(xxii)	$\mathsf{project}(\exists x F, S) \equiv \exists x\, \mathsf{project}(F, S)$
(xxiii)	$\mathsf{project}(\forall x F, S) \models \forall x\, \mathsf{project}(F, S)$

Table 4. Properties of Projection for $\mathcal{E}$-Formulas

(i)	$\mathsf{project}(E, \mathcal{L}_\mathcal{E}(E)) \equiv E$	(strengthens Tab. vii)
(ii)	$\mathcal{L}_\mathcal{E}(E) \subseteq S$ *if and only if* $\mathsf{project}(E, S) \equiv E$	(strengthens Tab. xiv)
(iii)	$\mathsf{project}(E, S) \equiv \mathsf{project}(E, \mathcal{L}_\mathcal{E}(E) \cap S)$	(strengthens Tab. xv)
(iv)	$\mathsf{forget}(E, S) \equiv \mathsf{forget}(E, \mathcal{L}_\mathcal{E}(E) \cap S)$	(strengthens Tab. xvi)
(v)	$F \models E$ *if and only if* $\mathsf{project}(F, \mathcal{L}_\mathcal{E}(E)) \models E$	(strengthens Tab. xvii)

universal first-order quantification (Tab. 3.xxi,xxiii). The precondition of the following theorem is sufficient for the converse of property Tab. 3.xxi. The essential part of its proof is deferred to Lemma 1 below. The theorem is based on the relation *linkless outside*, which applies to a pair of formulas and a literal scope if all ground atoms "involved in links" between the formulas (that is, are the atoms of complementary ground instances of two literals, one in each component of the pair) are contained in the literal scope, positively as well as negatively. For example, the pair of formulas $\langle p \vee q,\ \neg p \vee q \rangle$ is linkless outside the literal scope $\{+p, -p\}$. The relation is symmetric with respect to the pair components. Its formal definition is:

Definition 4 (Linkless Pair of Formulas). A pair of formulas $\langle F_1, F_2 \rangle$ is called *linkless outside* a literal scope S if and only if $\mathcal{L}(F_1) \cap \widetilde{\mathcal{L}(F_2)} \subseteq S \cap \widetilde{S}$.

Theorem 1 (Projection and Linkless Conjunction). *If F_1, F_2 are formulas and S is a literal scope such that $\langle F_1, F_2 \rangle$ is linkless outside S, then* $\mathsf{project}(F_1 \wedge F_2, S) \equiv \mathsf{project}(F_1, S) \wedge \mathsf{project}(F_2, S)$.

Proof. The case where $F_1 = F_2$ is trivial. Assume $F_1 \neq F_2$. Left-to-right is stated as Tab. 3.xxi. Right to left follows from Lemma 1 below, with $\Phi = \{F_1, F_2\}$, along with the semantic definitions of projection and conjunction (Tab. 1). □

Applications of Theorem 1. Boolean quantifier elimination is generalized by propositional projection computation, that is, computing for a propositional formula with the projection operator an equivalent formula without the projection operator. Theorem 1 can be applied to justify methods for this task. They take as input a formula $\mathsf{project}(F, S)$, where F is a propositional formula in negation normal form, and proceed as follows:

1. Compute a formula F' which is equivalent to F and has the property that for all conjunctive subformulas $(F_1 \wedge F_2)$ it holds that $\langle F_1, F_2 \rangle$ is linkless outside S.
2. Replace all literals in F' that are not in S by $\top$. The obtained formula is the result of projection computation.

Propositional formulas that satisfy the condition of step (1.) for the empty set as S (and thus also for any other set of literals as S) are called *linkless* [10]. Subclasses of linkless formulas are DNNF [5] and DNF, if complementary literals are not permitted in the same clause. Step (2.) is justified, since by property 3.xx and Theorem 1 the project operator in $\mathsf{project}(F', S)$ can be distributed inwards,

immediately in front of literal subformulas, where its value is determined by Tab. 3.xviii and xix. Regular tableaux, including semantic trees, whose nodes are labeled with literals, either propositional, or with just non-rigid variables, can be considered as representations of formulas – which are linkless. This is utilized by propositional tableau- and DPLL-based knowledge compilation methods that also perform Boolean quantifier elimination [10,15,12,16].

Another application of Theorem 1 is the justification of methods for propositional *Lit-simplifying* [11]. That is, computing for a given propositional formula an equivalent formula containing only literals in the essential literal base, which is assumed to be known. By Tab. 3.i, this task can be computed as described above for projection computation: transforming to an equivalent formula where all pairs of conjuncts are linkless outside the essential literal base, followed by replacing the literals not in the essential base by $\top$. There is also a dual alternative: From Tab. 3.i follows $\mathsf{project}(F, \mathcal{L}_{\mathcal{E}}(F)) \equiv \neg\mathsf{project}(\neg F, \mathcal{L}_{\mathcal{E}}(\neg F))$. Thus, *Lit-simplifying* can also be performed by operating on $\neg F$ instead of F, or, considered dually, by computing a formula that is equivalent to F and has the property that for all *disjunctive* subformulas $(F_1 \vee F_2)$ it holds that $\langle F_1, F_2\rangle$ is linkless outside the essential literal base of F (CNF formulas without tautological clauses, for example, have this property). The literals not in the essential base are then replaced by $\bot$.[3]

Conjunction and Essential Linklessness. Theorem 2, which follows, strengthens Theorem 1 for $\mathcal{E}$-formulas. In its precondition the property *essentially linkless outside* (Def. 5) takes the place of the stronger *linkless outside*. That *essentially linkless outside* is weaker follows from the fact that the essential literal base of a formula is a subset of its literal base. *Essentially linkless outside* is in a further respect different from *linkless outside*: it is independent of syntactic properties – if $\langle F_1, F_2\rangle$ is essentially linkless outside S, and F_1', F_2' are formulas such that $F_1' \equiv F_1$ and $F_2' \equiv F_2$, then also $\langle F_1', F_2'\rangle$ is essentially linkless outside S. This follows, since equivalent formulas have the same essential literal base.

Definition 5 (Essentially Linkless Pair of Formulas). A pair of formulas $\langle F_1, F_2\rangle$ is called *essentially linkless outside* a literal scope S if and only if $\mathcal{L}_{\mathcal{E}}(F_1) \cap \widetilde{\mathcal{L}_{\mathcal{E}}(F_2)} \subseteq S \cap \widetilde{S}$.

Example 4 (Essentially Linkless Pair of Formulas). Let S be the set of literals $\{+p, -p\}$. Then $\langle p \vee (q \wedge \neg q),\ \neg p \vee q\rangle$ is not linkless outside S, but essentially linkless outside S.

[3] In [11, Sect. 3.2] it is erroneously stated that Lit-simplifying of a propositional formula in *negation normal form (NNF)* can be performed by (in our terminology) substituting the literals which are not in the essential literal base with $\bot$, and thus would be a polynomial operation. The statement is false: Let $F \stackrel{\text{def}}{=} (p \vee \neg p)$. Then F is in NNF and $\mathcal{L}_{\mathcal{E}}(F) = \emptyset$. Substituting in F the literals not in $\mathcal{L}_{\mathcal{E}}(F)$ with $\bot$ yields $(\bot \vee \bot)$, which is not equivalent to F.

Theorem 2 (Projection and Essentially Linkless Conjunction). *If F_1, F_2 are $\mathcal{E}$-formulas and S is a literal scope such that $\langle F_1, F_2\rangle$ is essentially linkless outside S, then* $\mathsf{project}(F_1 \wedge F_2, S) \equiv \mathsf{project}(F_1, S) \wedge \mathsf{project}(F_2, S)$.

Proof. Can be shown in the same way as Theorem 1, but based on a variant of Lemma 1, where Φ is a set of $\mathcal{E}$-formulas, and *linkless outside* in (A1) is replaced by *essentially linkless outside*. The varied lemma can be proven like the original one, with $\mathcal{L}_\mathcal{E}$ in place of $\mathcal{L}$ and referring to Def. 3 instead of Prop. 1. □

Universal Quantification and Linklessness. The same principles that permit to push the projection operator inside conjunctions can be applied to universal quantification. We state it for *linkless outside* as precondition in the following theorem.

Theorem 3 (Projection and Linkless Universal Quantification). *If F is a formula, x is a variable that occurs free or not at all in F, and S is a literal scope such that for all $t, u \in \mathsf{GTERMS}$ where $t \neq u$ it holds that $\langle F\{x \mapsto t\}, F\{x \mapsto u\}\rangle$ is linkless outside S, then* $\mathsf{project}(\forall x F, S) \equiv \forall x\, \mathsf{project}(F, S)$.

Proof. The case where x does not occur in F is trivial. Assume x occurs free in F. Left-to-right is stated as Tab. 3.xxiii. Right-to-left follows from Lemma 1 below, with $\Phi = \{F\{x \mapsto t\} \mid t \in \mathsf{GTERMS}\}$, along with the semantic definitions of projection and universal quantification (Tab. 1). □

The Lemma Underlying Theorems 1–3. We conclude this section by stating the lemma underlying Theorems 1–3 and giving a proof sketch for it.

Lemma 1. *If $\langle I, \beta\rangle$ is an interpretation, S is a literal scope, and Φ a set of formulas such that*

(A1) for all formulas $F, G \in \Phi$ such that $F \neq G$ it holds that $\langle F\beta, G\beta\rangle$ is linkless outside S, and

(A2) for all formulas $F \in \Phi$ it holds that $\langle I, \beta\rangle \models \mathsf{project}(F, S)$,

then there exists an interpretation $\langle J, \beta\rangle$ such that

(C1) for all formulas $F \in \Phi$ it holds that $\langle J, \beta\rangle \models F$, and

(C2) $J \cap S \subseteq I$.

Proof (Sketch). Let $\langle I, \beta\rangle, S$, and Φ be as specified for the lemma, and assume that they satisfy preconditions (A1) and (A2). For all $F \in \Phi$ let J_F be a structure such that $\langle J_F, \beta\rangle \models F$ and $J_F \cap S \subseteq I$. The existence of such J_F follows from (A2) and the semantic definition of project. We prove the lemma by showing the construction of a structure J such that consequences (C1) and (C2) are satisfied. The construction of J is based on two auxiliary sets of literals, $\mathcal{L}^{+A}$ and $\mathcal{L}^{-A}$, which are associated with each ground atom A:

$$\mathcal{L}^{+A} \stackrel{\text{def}}{=} \bigcup_{+A \in J_F,\ F \in \Phi} \mathcal{L}(F\beta), \qquad \mathcal{L}^{-A} \stackrel{\text{def}}{=} \bigcup_{-A \in J_F,\ F \in \Phi} \mathcal{L}(F\beta).$$

The structure J is then defined by:

$$\text{If } +A \notin \mathcal{L}^{+A} \text{ and } (-A \in \mathcal{L}^{-A} \text{ or } -A \in I), \text{ then } -A \stackrel{\text{def}}{\in} J, \text{ else } +A \stackrel{\text{def}}{\in} J.$$

For all elements F of Φ let $M_F \stackrel{\text{def}}{=} J_F \cap \widetilde{J}$. Then $J = J_F[\widetilde{M_F}]$. It can be verified that $M_F \cap \mathcal{L}(F\beta) = \emptyset$. Since $\langle J_F, \beta\rangle \models F$, consequence (C1) then follows from Prop. 1. It can be verified that $J \subseteq \bigcup_{F\in\Phi} J_F \cup I$. Consequence (C2) then follows since $J_F \cap S \subseteq I$. □

5 Unlinked Literal Occurrences

The following theorem uses the *linkless outside* property related to specific occurrences of literals in formulas. If a literal is "not linked" to its context within a formula, then replacing the literal by $\top$ yields a formula whose projection is equivalent to the projection of the original one. The idea is to use the equivalence stated in the theorem as building block for methods to eliminate the projection operator, although this remains largely future work. As a first approach, we show below, with Prop. 2, that a restricted variant of the *Ackermann lemma* [17,18], the basis of several known methods for second-order quantifier elimination [3,9,8], can be modeled with the theorem.

Following the terminology of [19], if F is a formula with a subformula occurrence replaced by a *hole*, and G is a formula, then $F[G]$ is F with the hole replaced by G. At the same time $F[G]$ indicates that the formula F contains an occurrence of the subformula G.

Theorem 4 (Eliminating an Unlinked Literal Occurrence). *If S is a literal scope, L is a literal of which no instance is in S, and $F[L]$ is a first-order formula such that*

(A1) *for all subformulas $(F_1[L] \wedge F_2)$ and $(F_2 \wedge F_1[L])$ of $F[L]$ it holds that $\langle L,\ F_2\rangle$ is linkless outside S, and*

(A2) *for all subformulas $(\forall x F'[L])$ of $F[L]$ and $t_1, t_2 \in \mathsf{GTERMS}$ such that $t_1 \neq t_2$ it holds that $\langle L\{x \mapsto t_1\},\ F'[L]\{x \mapsto t_2\}\rangle$ is linkless outside S,*

then

$$\mathsf{project}(F[L], S) \equiv \mathsf{project}(F[\top], S).$$

Proof (Sketch). The left-to-right direction follows from Tab. 3.ii, since $F[L] \models F[\top]$. The right-to-left direction can be shown as follows: Let $\langle I, \beta\rangle$ be an interpretation such that $\langle I, \beta\rangle \models \mathsf{project}(F[\top], S)$. By the definition of project, there exists an interpretation $\langle J, \beta\rangle$ such that $\langle J, \beta\rangle \models F[\top]$ and $J \cap S \subseteq I$. We need to show that $\langle I, \beta\rangle \models \mathsf{project}(F[L], S)$, that is, that there exists an interpretation $\langle K, \beta\rangle$ such that $\langle K, \beta\rangle \models F[L]$ and $K \cap S \subseteq I$. In case $\langle J, \beta\rangle \models F[L]$, we have found in J a suitable K. The other case, where $\langle J, \beta\rangle \not\models F[L]$, can be shown by induction on first-order formulas $F[L]$, where L is a literal, and $F[L]$ satisfies (A1) and (A2). Before stating the induction property, we need some more notation: If F, G are formulas and β is a variable assignment, then $F(\beta \downarrow G)$ denotes

F with those variables that are free in G replaced by their images in β. The induction property is: *If S is a literal scope such that no instance of L is in S and $\langle J, \beta\rangle$ is an interpretation such that $\langle J, \beta\rangle \models F[\top]$ and $\langle J, \beta\rangle \not\models F[L]$, then there exists a set M of ground instances of $L(\beta \downarrow F[L])$ such that $\langle J[M], \beta\rangle \models F[L]$.* The set of literals M is defined such that $S \cap M = \emptyset$, which allows to conclude $J[M] \cap S \subseteq J \cap S$. If $K = J[M]$, then from $J \cap S \subseteq I$ follows $K \cap S \subseteq I$.

We sketch the base case for literals, and the induction steps for conjunction and universal quantification. The remaining induction steps for disjunction and existential quantification are straightforward to show. In the base case where $F[L] = L$ it holds that $\langle J, \beta\rangle \not\models L$. A suitable M is $\{L\beta\}$. The induction step for $F[L] = (F_1[L] \wedge F_2)$ can be shown as follows: From $\langle J, \beta\rangle \models (F_1[\top] \wedge F_2)$ and $\langle J, \beta\rangle \not\models (F_1[L] \wedge F_2)$ follows $\langle J, \beta\rangle \not\models F_1[L]$, hence by the induction assumption, there exists a set M of instances of $L(\beta \downarrow F_1[L])$ such that $\langle J[M], \beta\rangle \models F_1[L]$. By condition (A1) it holds that $\mathcal{L}(F_2) \cap \widetilde{M} = \emptyset$. From $\langle J, \beta\rangle \models F_2$, then by Prop. 1 follows $\langle J[M], \beta\rangle \models F_2$.

The induction step for $F[L] = (\forall x F')$ can be shown as follows: From $\langle J, \beta\rangle \models \forall x\ F'[\top]$ follows that for all ground terms t it holds that $\langle J, \beta\frac{t}{x}\rangle \models F'[\top]$. Let T be the set of ground terms t such that $\langle J, \beta\frac{t}{x}\rangle \not\models F'[L]$. From $\langle J, \beta\rangle \not\models \forall x\ F'[L]$ follows that T is not empty. For all $t \in T$ let M_t be a set of literals, ground instances of $L(\beta\frac{t}{x} \downarrow F'[L])$ such that $\langle J[M_t], \beta\frac{t}{x}\rangle \models F'[L]$. The existence of these M_t follows from the induction assumption. Let $M \stackrel{\text{def}}{=} \bigcup_{t\in T} M_t$. The induction conclusion $\langle J[M], \beta\rangle \models \forall x F'[L]$ is implied by the fact that for all ground terms t it holds that $\langle J[M], \beta\frac{t}{x}\rangle \models F'[L]$, which can be shown as follows: Let t be an element of T and s be an arbitrary ground term, different from t. By (A2) it holds that $\widetilde{\mathcal{L}(L\{x \mapsto t\})} \cap \mathcal{L}(F'[L]\{x \mapsto s\}) = \emptyset$. Since $M_t \subseteq \mathcal{L}(F'[L]\{x \mapsto t\})$, it follows that

$$\widetilde{M_t} \cap \mathcal{L}(F'[L]\{x \mapsto s\}) = \emptyset. \tag{i}$$

If $t \in T$, then $\langle J[M_t], \beta\frac{t}{x}\rangle \models F'[L]$, hence $\langle J[M_t], \beta\frac{t}{x}\rangle \models F'[L]\{x \mapsto t\}$. From Eq. (i) follows $(M - M_t) \cap \mathcal{L}(F'[L]\{x \mapsto t\}) = \emptyset$. By Prop. 1 follows $\langle J[M], \beta\frac{t}{x}\rangle \models F'[L]\{x \mapsto t\}$, hence $\langle J[M], \beta\frac{t}{x}\rangle \models F'[L]$. Else, if $t \notin T$, from Eq. (i) follows $M \cap \mathcal{L}(F'[L]\{x \mapsto t\}) = \emptyset$. From $\langle J, \beta\frac{t}{x}\rangle \models F'[L]$ it can be concluded, similarly as in the case where $t \in T$, with Prop. 1 that $\langle J[M], \beta\frac{t}{x}\rangle \models F'[L]$. □

Application of Theorem 4. The Ackermann lemma states an equivalence of formulas of a certain form and with a second-order quantifier to first-order formulas. Proposition 2 can be used to prove a restricted variant of the Ackermann lemma (the restriction is that the subformula which may contain multiple instantiated occurrences of the quantified predicate is not permitted to contain the existential first-order quantifier). The proposition statement shows that equivalence with respect to the projection (expressed conveniently as forgetting) is preserved by eliminating a single occurrence of a positive literal with predicate p. The way the literal occurrence is eliminated is identical to the way in which according to the Ackermann lemma all occurrences of literals with positive p are eliminated. By iterated rewriting with the equivalence of Prop. 2, all positive occurrences of p can be eliminated, permitting the negative occurrence finally to

be eliminated directly according to Theorem 4, followed by dropping the forget operator. The result is then the same first-order formula that would be obtained by a single rewriting step with the Ackermann lemma. Of course, as for the Ackermann lemma, there is also a dual variant of Prop. 2 with polarities of the occurrences of p switched, but we do not state this explicitly here.

We use additional notation: A sequence of terms $t_1, ..., t_n$ is abbreviated by $\overline{t}$. If F is a formula, then $F(\overline{t})$ denotes F with all occurrences of variables x_i replaced by term t_i, for $i \in \{1, ..., n\}$.

Proposition 2 (Ackermann Lemma Step for Universal Formulas). *Let p be a n-ary predicate symbol, and F, G be first-order formulas such that p does not occur in F and does occur in G only in positive literals, G has the form $G[p(\overline{t})]$, and G does not contain an existential quantifier. Then*

$$\begin{array}{ll} \textit{(F1)} & \mathsf{forget}((\forall \overline{x}\ \neg p(\overline{x}) \vee F(\overline{x})) \wedge G[p(\overline{t})],\ P_+) \\ \textit{(F2)} & \mathsf{forget}((\forall \overline{x}\ \neg p(\overline{x}) \vee F(\overline{x})) \wedge G[F(\overline{t})],\ P_+), \end{array} \quad \equiv$$

where P_+ is the set of all positive ground literals with predicate p.

Proof (Sketch – see [16] for more details). Formula (F1) is equivalent to the following formula (F1′), since the formula arguments in both forget expressions are equivalent.

$$\text{(F1}') \quad \mathsf{forget}(\forall \overline{x}\ (p(\overline{x}) \wedge F(\overline{x}) \wedge G[\top]) \vee (\neg p(\overline{x}) \wedge G[\overline{x} \neq \overline{t}]), P_+)$$

The leftmost literal $p(\overline{x})$ in (F1′) meets the requirements on L in Theorem 4. By that theorem (F1′) is equivalent to (F2′), which, again by equivalence of the formula arguments in both forget expressions, is equivalent to (F2).

$$\text{(F2}') \quad \mathsf{forget}(\forall \overline{x}\ (\top \wedge F(\overline{x}) \wedge G[\top]) \vee (\neg p(\overline{x}) \wedge G[\overline{x} \neq \overline{t}]), P_+)$$

□

6 Conclusion

We expect that the consideration of polarity will play an important role in some applications of predicate quantifier elimination – for example to compute knowledge base extracts that just keep information about a predicate in one polarity and thus suffice to answer queries containing the predicate just in this polarity, or for knowledge base modularization, where it should be ensured that additions to a knowledge base affect some predicate only in a certain polarity. We presented a formalization that expresses polarity sensitive predicate quantification in an easily accessible way by means of literal sets. It applies to first-order logic, and thus should provide a basis also for other logics used in knowledge representation that are more expressive than just propositional logic. We applied the formalization to show some properties of literal projection which relate to methods for predicate quantifier elimination. These properties already suffice as building blocks to justify some methods that can in practice be used for predicate quantifier elimination, and provide a basis to extend the successful applications of projection computation in the context of propositional knowledge compilation

to more expressive logics. The investigation of further methods, especially for non-propositional formulas, remains future work.

Acknowledgments. I am grateful to Renate A. Schmidt for valuable comments and discussions on earlier versions of some of the material in the paper, and to anonymous referees for helpful suggestions to improve the presentation.

References

1. Gabbay, D., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. In: KR 1992, pp. 425–435 (1992)
2. Lin, F., Reiter, R.: Forget It! In: Working Notes, AAAI Fall Symposium on Relevance, pp. 154–159 (1994)
3. Doherty, P., Łukaszewicz, W., Szałas, A.: Computing circumscription revisited: A reduction algorithm. J. Autom. Reason. 18(3), 297–338 (1997)
4. McMillan, K.L.: Applying SAT methods in unbounded symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 250–264. Springer, Heidelberg (2002)
5. Darwiche, A.: Decomposable negation normal form. JACM 48(4), 608–647 (2001)
6. Wernhard, C.: Semantic knowledge partitioning. In: Alferes, J.J., Leite, J.A. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 552–564. Springer, Heidelberg (2004)
7. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: KR 2006, pp. 187–197 (2006)
8. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publications (2008)
9. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic. I. the core algorithm SQEMA. Log. Meth. in Comp. Science 2(1-5), 1–26 (2006)
10. Murray, N.V., Rosenthal, E.: Tableaux, path dissolution and decomposable negation normal form for knowledge compilation. In: Cialdea Mayer, M., Pirri, F. (eds.) TABLEAUX 2003. LNCS, vol. 2796, pp. 165–180. Springer, Heidelberg (2003)
11. Lang, J., Liberatore, P., Marquis, P.: Propositional independence — formula-variable independence and forgetting. JAIR 18, 391–443 (2003)
12. Wernhard, C.: Automated Deduction for Projection Elimination. PhD thesis, Universität Koblenz-Landau, Germany (to appear, 2008)
13. Urban, J.: MPTP 0.2: Design, implementation, and initial experiments. J. Autom. Reason. 37(1-2), 21–43 (2006)
14. Ebbinghaus, H.D., Flum, J., Thomas, W.: Einführung in die mathematische Logik, 4th edn. Spektrum Akademischer Verlag, Heidelberg (1996)
15. Huang, J., Darwiche, A.: DPLL with a trace: From SAT to knowledge compilation. In: IJCAI 2005, pp. 156–162 (2005)
16. Wernhard, C.: Literal projection for first-order logic (extended version). Technical Report Fachbereich Informatik, Universität Koblenz-Landau (to appear, 2008)
17. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Math. Annalen 110, 390–413 (1935)
18. Szalas, A.: On the correspondence between modal and classical logic: An automated approach. J. Log. Comput. 3(6), 605–620 (1993)
19. Dershowitz, N., Plaisted, D.A.: Rewriting. In: Handbook of Automated Reasoning, vol. I, pp. 537–610. Elsevier, Amsterdam (2001)

Meta Level Reasoning and Default Reasoning*

Yi Zhou and Yan Zhang

Intelligent Systems Lab
University of Western Sydney
Locked Bag 1797, NSW, Australian

Abstract. In this paper, we propose a logic framework for meta level reasoning as well as default reasoning in a general sense, based on an arbitrary underlying logic. In this framework, meta level reasoning is the task of how to deduce new meta level rules by giving a set of rules, whilst default reasoning is the problem of what are the possible candidate beliefs by giving them. We define the semantics for both meta level reasoning and default reasoning and investigate their relationships. We show that this framework captures various nonmonotonic paradigms, including answer set programming, default logic, contextual default reasoning, by applying the underlying logic to different classes. Finally, we show that this framework can be reduced into answer set programming.

1 Introduction

Consider that an agent A is reasoning about a system S, where information can be captured by a logic consisting of a language $\mathcal{L}$ and an entailment relation $\models_{\mathcal{L}}$ among formulas in $\mathcal{L}$. In principle, if the agent A has perfect reasoning power, then its information about S should be a set of formulas in $\mathcal{L}$ closed under $\models_{\mathcal{L}}$, say a candidate belief. Suppose that Γ is a set of formulas, representing the information that A considers to be true about S. Thus, $Cn(\Gamma)$, the closure of Γ under $\models_{\mathcal{L}}$, should be included in every possible candidate beliefs.

However, $Cn(\Gamma)$ is not the only information that A can have about S. More can be obtained by meta level rules, which represent statements about possible candidate beliefs in a meta level language. For instance, a statement may claim that "if a candidate belief does not contain F_1, then it must contain F_2". In fact, the well-known closed world assumption is a special case of this statement when F_2 is $\neg F_1$, providing that the language $\mathcal{L}$ has the connective $\neg$ to represent negative information in the system.

Meta level rules cannot be represented in the language $\mathcal{L}$ itself since the objects they deal with are not formulas in $\mathcal{L}$ but statements about candidate beliefs. More precisely, meta level rules are composed by primitive statements and meta level connectives. The former are sentences stating whether a formula is contained in a possible candidate belief, while the latter are words connecting those primitive statements in a meta language. Consider the example mentioned above. There are two primitive statements, namely, "the candidate belief contains F_1" and "the candidate belief contains F_2". Furthermore, they are connected by two meta level connectives, namely "not" and "if then".

* This work is supported by Australian Research Council (ARC) Discovery Projects grant DP0666540.

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 403–415, 2008.

Hence, the problem of how to represent meta level rules can be divided into two parts, how to represent primitive statements and how to represent meta level connectives. In this paper, we simply write a formula F in $\mathcal{L}$ to represent the primitive statement "the candidate belief contains F". On the other hand, we adopt a set of propositional meta level connectives, including rule and ($\&$), rule or ($\wr$), rule negation ($\sim$) and rule implication ($\Rightarrow$). For example, the meta level statement in the above example can be represented as $\sim F_1 \Rightarrow F_2$.

There are two fundamental reasoning tasks in relation to reasoning about meta level rules. The first one is called meta level reasoning. That is, which meta level rules can be deduced by giving a set of rules. Another reasoning task is default reasoning, which is the problem of what are the possible candidate beliefs by giving a set of rules.

In this paper, we propose a logic framework for both meta level reasoning and default reasoning in a general sense, based on an arbitrary underlying logic, which consists of a language $\mathcal{L}$ and an entailment relation $\models_{\mathcal{L}}$ under some restrictions. In this sense, there are numerous instances of the underlying logic, such as set inclusion, propositional logic, epistemic logic and so on.

The reasons why we consider arbitrary underlying logics are threefold. Firstly, due to diversity of applications, the logic for representing the system S may vary from the simplest one to more complicated ones. Secondly, considering the two reasoning tasks in a general sense may help us to reveal the nature of them. Finally, a general framework not only unifies a number of existing approaches but also initiates promising ones.

The rest of this paper is organized as follows. Next, we propose the syntax and basic semantics of the logic framework. In Section 3, we define both meta level reasoning and default reasoning of the framework semantically, and investigate their relationships. Then, we show that this framework is powerful enough to capture various existing approaches and possible new ones in Section 4. In Section 5, we show that it can be reduced into its simplest case, namely answer set programming. Finally, we draw our conclusions.

2 Syntax and Basic Semantics

To begin with, we need to specify what a logic is. We adopt Gentzen's idea [1] of standard logic system, which consists of two components. Firstly, it has a syntax, namely, a formal language to define what are the objects dealt with in this logic system. We denote it by a language $\mathcal{L}$. Basically, it can be represented as a set. Elements in $\mathcal{L}$ are called *formulas*. Secondly, the logic system should have reasoning ability, that is, to answer the question whether a formula can be derived by other formulas. This is formalized by an *entailment relation* $\models_{\mathcal{L}}$ between a set of formulas and a formula in $\mathcal{L}$. In other words, $\models_{\mathcal{L}}$ is a relation $\models_{\mathcal{L}} \subseteq 2^{\mathcal{L}} \times \mathcal{L}$, that satisfies the following two restrictions:

Reflexivity if $F \in \Gamma$, then $\Gamma \models_{\mathcal{L}} F$;
Transitivity (cut) if for all $F' \in \Gamma'$, $\Gamma \models_{\mathcal{L}} F'$, then $\Gamma' \models_{\mathcal{L}} F$ implies that $\Gamma \models_{\mathcal{L}} F$,

where $\Gamma, \Gamma' \subseteq \mathcal{L}$, $F, F' \in \mathcal{L}$.

According to reflexivity and transitivity, a logic system also satisfies the following properties.

Proposition 1. *Let $\mathcal{L}$ be a language and $\models_{\mathcal{L}}$ the corresponding entailment relation satisfying reflexivity and transitivity. Then, it also satisfies the following properties:*

Monotonicity *if $\Gamma \subseteq \Gamma'$, then for all $F \in \mathcal{L}$ such that $\Gamma \models_{\mathcal{L}} F$, $\Gamma' \models_{\mathcal{L}} F$;*
Equivalency *if for all $F' \in \Gamma'$, $\Gamma \models_{\mathcal{L}} F'$ and for all $F \in \Gamma$, $\Gamma' \models_{\mathcal{L}} F$, then for all $G \in \mathcal{L}$, $\Gamma \models_{\mathcal{L}} G$ iff $\Gamma' \models_{\mathcal{L}} G$;*
Extendability *if for all $F' \in \Gamma'$, $\Gamma \models_{\mathcal{L}} F'$, then for all $F \in \mathcal{L}$, $\Gamma \cup \Gamma' \models_{\mathcal{L}} F$ iff $\Gamma \models_{\mathcal{L}} F$.*

Of course, classical propositional logic is a typical example of such a logic. There are numerous other examples, such as first order logic, modal logic, probabilistic logic, intuitionistic logic and so on. In particular, set inclusion can also be considered as a logic. Let $Atom$ be a set of atoms. The formulas in the language $\mathcal{L}$ of set inclusion are defined as elements in $Atom$, and the entailment relation between a set Γ of formulas (i.e. a subset of $Atom$) and a formula F (i.e. an element in $Atom$) is defined as set inclusion (i.e. $\Gamma \models_{\mathcal{L}} F$ iff $F \in \Gamma$). It is obvious that this entailment relation (i.e. set inclusion) satisfies reflexivity and transitivity.

However, Reiter's default logic is not a logic according to this definition if the entailment relation is defined as credulous reasoning or skeptical reasoning. One reason is that both credulous reasoning and skeptical reasoning do not satisfy transitivity. Another reason is that the consequence of both credulous reasoning and skeptical reasoning is not a default rule but a propositional formula[1].

A *closure* is a set C of formulas in $\mathcal{L}$ closed under the entailment relation $\models_{\mathcal{L}}$. That is, C is a closure iff for all $F \in \mathcal{L}$ such that $C \models_{\mathcal{L}} F$, $F \in C$. By reflexivity, it is easy to see that if $C \not\models_{\mathcal{L}} F$[2], then $F \notin C$. Hence, $C \models_{\mathcal{L}} F$ iff $F \in C$. It is easy to see that if C_1 and C_2 are two closures, then so is $C_1 \cap C_2$.

Proposition 2. *Let $\mathcal{L}$ be a language and $\models_{\mathcal{L}}$ the corresponding entailment relation satisfying reflexivity and transitivity. Let Γ be a set of formulas in $\mathcal{L}$. There exists a unique closure C such that for all $F \in \mathcal{L}$, $\Gamma \models_{\mathcal{L}} F$ iff $C \models_{\mathcal{L}} F$.*

We write $Cn(\Gamma)$ to denote this closure of Γ. For convenience, we simply use Γ to denote $Cn(\Gamma)$ if it is clear from the context. Clearly, if $\Gamma_1 \subseteq \Gamma_2$, then $Cn(\Gamma_1) \subseteq Cn(\Gamma_2)$.

Based on the underlying logic $\mathcal{L}$, we define a meta level language $\mathcal{ML}(\mathcal{L})$, following a similar construction of general default logic [3]. One major difference is that, instead of classical propositional logic, we use an arbitrary underlying logic as discussed above.

The meta level language $\mathcal{ML}(\mathcal{L})$ is defined upon $\mathcal{L}$ by introducing a set of *meta level rule connectives* (*rule connectives* for short), including *rule and* ($\&$), *rule or* ($\wr$), *rule implication* ($\Rightarrow$), and a special 0-ary connective *falsity* $\bot$ as follows:

$$R := F \mid \bot \mid R \,\&\, R \mid R \wr R \mid R \Rightarrow R,$$

[1] Hence, our definition of logic is not the same as Brewka and Eiter's [2]. According to their definition, both default logic and answer set programming are logics.
[2] We write $\Gamma \not\models_{\mathcal{L}} F$ if it is not the case that $\Gamma \models_{\mathcal{L}} F$, the same for other similar notations used later.

where $F \in \mathcal{L}$. We also introduce other rule connectives *truth* $\top$, *rule negation* $\sim$, and *rule equivalence* $\Leftrightarrow$. $\top$, $\sim R$ and $R_1 \Leftrightarrow R_2$ are considered as shorthand of $\bot \Rightarrow \bot$, $R \Rightarrow \bot$ and $(R_1 \Rightarrow R_2)\ \&\ (R_2 \Rightarrow R_1)$ respectively. Formulas in $\mathcal{ML}(\mathcal{L})$ are called *meta level rules* (*rules* for short). In particular, formulas in $\mathcal{L}$ are also rules. For convenience, we call them *facts*. A *rule base* is a set of rules. The *subrule* relationship between two rules are defined recursively.

- R_1 is a subrule of R_1.
- Both R_1 and R_2 are subrules of $R_1\ \&\ R_2$, $R_1 \wr R_2$ and $R_1 \Rightarrow R_2$.

In particular, if F is a fact and also a subrule of R, we say that F is a *subfact* of R.

In the basic semantics, we define the *satisfaction relation* $\models_B$ between closures in the underlying language $\mathcal{L}$ and meta level rules recursively as follows:

- If R is a fact, then $C \models_B R$ iff $C \models_{\mathcal{L}} R$;
- $C \not\models_B \bot$;
- $C \models_B R\ \&\ S$ iff $C \models_B R$ and $C \models_B S$;
- $C \models_B R \wr S$ iff $C \models_B R$ or $C \models_B S$;
- $C \models_B R \Rightarrow S$ iff $C \not\models_B R$ or $C \models_B S$.

Thus, $C \models_B \top$. $C \models_B \sim R$ iff $C \models_B R \Rightarrow \bot$ iff $C \not\models_B R$ or $C \models_B \bot$ iff $C \not\models_B R$. $C \models_B R \Leftrightarrow S$ iff $C \models_B (R \Rightarrow S)\ \&\ (S \Rightarrow R)$ iff $C \models_B R \Rightarrow S$ and $C \models_B S \Rightarrow R$ iff (a) $C \models_B R$ and $C \models_B S$ or (b) $C \not\models_B R$ and $C \not\models_B S$. We say that C *satisfies* R, also C is a *model* of R iff $C \models_B R$. We say that two rules are *weakly equivalent* if they have the same set of models. We say that C satisfies a rule base Δ iff C satisfies all rules in Δ.

Example 1. Consider the rule $\sim F_1 \Rightarrow F_2$. If a closure contains neither F_1 nor F_2, then it is not a model of this rule. On the other hand, a closure containing F_1 satisfies this rule, so does a closure containing F_2.

Note that the underlying logic may have internal relationships among formulas. For instance, consider a rule $F_1\ \&\ \sim F_2$. If in an underlying logic $\mathcal{L}$, $\{F_1\} \models_{\mathcal{L}} F_2$, then there is no model of the rule $F_1\ \&\ \sim F_2$. However, if in another underlying logic $\mathcal{L}'$, $\{F_1\} \not\models_{\mathcal{L}'} F_2$, then a closure containing F_1 but not F_2 is a model of the rule $F_1\ \&\ \sim F_2$.

The basic semantics can be translated into classical propositional logic. Let $At(\mathcal{L})$ be a set of atoms in propositional logic and At a one-to-one mapping from $\mathcal{L}$ to $At(\mathcal{L})$. Given a meta level rule R, by $Tr_{CL}(R)$ we denote the propositional formula obtained from R by simultaneously replacing every subfact F in R with $At(F)$ and every rule connective with corresponding classical propositional connectives. Given a closure C, by $At(C)$, we denote the propositional assignment[3] over $At(\mathcal{L})$ such that $F \in C$ iff $At(F) \in At(C)$.

Theorem 1. *Let R be a rule and C a closure. $C \models_B R$ iff $At(C)$ is a model of $Tr_{CL}(R)$ in classical propositional logic.*

Corollary 1. *Let R_1 and R_2 be two rules. If $Tr_{CL}(R_1)$ is equivalent to $Tr_{CL}(R_2)$ in classical propositional logic, then for all closures C, $C \models_B R_1$ iff $C \models_B R_2$.*

[3] We identify a propositional assignment as the set of atoms assigned to be true in it.

3 Meta Level Reasoning and Default Reasoning

A natural question is so-called meta level reasoning, namely, how to derive new meta level rules by giving a set of rules. We use a bi-level semantics for this reasoning task. The semantics is originated from the logic of here-and-there, which was developed by Heyting and adopted by Pearce [4] for answer set programming.

A *bi-level interpretation* in $\mathcal{ML}(\mathcal{L})$ is a pair $\langle C_1, C_2\rangle$, where C_1 and C_2 are both closures in $\mathcal{L}$. The satisfaction relation $\models_{BI}$ between bi-level interpretations and meta level rules is defined recursively as follows:

- if R is a fact, then $\langle C_1, C_2\rangle \models_{BI} R$ iff $C_1 \models_B R$ and $C_2 \models_B R$;
- $\langle C_1, C_2\rangle \not\models_{BI} \bot$;
- $\langle C_1, C_2\rangle \models_{BI} R_1 \,\&\, R_2$ iff $\langle C_1, C_2\rangle \models_{BI} R_1$ and $\langle C_1, C_2\rangle \models_{BI} R_2$;
- $\langle C_1, C_2\rangle \models_{BI} R_1 \wr R_2$ iff $\langle C_1, C_2\rangle \models_{BI} R_1$ or $\langle C_1, C_2\rangle \models_{BI} R_2$;
- $\langle C_1, C_2\rangle \models_{BI} R_1 \Rightarrow R_2$ iff
  1. $\langle C_1, C_2\rangle \not\models_{BI} R_1$ or $\langle C_1, C_2\rangle \models_{BI} R_2$ and
  2. $C_2 \models_B R_1 \Rightarrow R_2$.

We say that $\langle C_1, C_2\rangle$ is a *bi-level model* of R iff $\langle C_1, C_2\rangle \models_{BI} R$. We say that a rule base Δ *implies* a rule R, denoted by $\Delta \models_{BI} R$, iff all bi-level models of Δ are bi-level models of R as well.

The reason why we call this semantics bi-level is that the two components of the pair represent two levels of information respectively. The second lies on the underlying level, which represents a possible guess of the agent about the system, while the first one lies on the meta level, which represents the actual set of information that the agent can have by fixing the underlying level information.

Example 2 (Example 1 continued). Consider the rule $\sim F_1 \Rightarrow F_2$. Suppose that C_0 is the closure $TH(\emptyset)$, while C_1 is a closure containing F_1. Then, $\langle C_1, C_1\rangle$ is a bi-level model of $\sim F_1 \Rightarrow F_2$, so is $\langle C_0, C_1\rangle$. However, $\langle C_1, C_0\rangle$ and $\langle C_0, C_0\rangle$ are not. Thus, $\{\sim F_1 \Rightarrow F_2\} \not\models_{BI} F_1 \wr F_2$ since $\langle C_0, C_1\rangle$ is a bi-level model of $\sim F_1 \Rightarrow F_2$ but not a bi-level model of $F_1 \wr F_2$. However, one can check that $\{F_1 \wr F_2\} \models_{BI} \sim F_1 \Rightarrow F_2$ no matter what the underlying logic is.

The bi-level semantics and the basic semantics are closely related. By induction on the structure of R, we have the following result.

Proposition 3. *Let $\langle C_1, C_2\rangle$ be a bi-level interpretation and R a rule.*

- $\langle C_1, C_2\rangle \models_{BI} \sim R$ *iff* $C_2 \models_B \sim R$.
- $\langle C_1, C_1\rangle \models_{BI} R$ *iff* $C_1 \models_B R$.
- *If* $\langle C_1, C_2\rangle \models_{BI} R$, *then* $C_2 \models_B R$.

The result of Proposition 3 is not new; it holds for the logic of here-and-there as well [5]. In fact, the bi-level semantics shares the nature of the logic here-and-there. There are two major differences. Firstly, the bi-level semantics is generalized into an arbitrary case, whilst the logic of here-and-there is only concerned with set inclusion (i.e. atom sets). Secondly, in the bi-level semantics, we do not require the restriction that the first component of the pair has to be a subset of the second one.

Proposition 4. *Let Δ be a rule base and R_1 and R_2 two rules. $\Delta \cup \{R_1\} \models_{BI} R_2$ iff $\Delta \models_{BI} R_1 \Rightarrow R_2$.*

Proposition 5. *Let Δ be a rule base and R_1 and R_2 two rules. If $\Delta \models_{BI} R_2$, then $\Delta \cup \{R_1\} \models_{BI} R_2$.*

Proposition 6. *Let Δ be a rule base, R a rule and C a closure. If $C \models_B \Delta$ and $\Delta \models_{BI} R$, then $C \models_B R$.*

Proposition 7. *Let $\langle C_1, C_2\rangle$ be a bi-level interpretation and R a rule. $\langle C_1, C_2\rangle \models_{BI} R$ iff $\langle C_1 \cap C_2, C_2\rangle \models_{BI} R$.*

According to Proposition 7, the bi-level semantics, when the underlying logic is set inclusion, is indeed identical to the logic of here-and-there. The reasons why we make this minor change (i.e. to remove the restriction) are twofold. On the one hand, the restriction seems unnecessary and not natural from a mathematical point of view. On the other hand, the intuitions behind the bi-level semantics without the restriction are clearer than that with it.

Perhaps, another reasoning task is more interesting, namely default reasoning, which is the problem of what are the possible candidate beliefs by giving a set of meta level rules. We introduce two semantics for default reasoning. One is a reduction style extension semantics, following the idea from Ferraris' work [6] on answer set semantics for so-called propositional theories, and extended to general default logic by Zhou et al. [3]. The other is equilibrium semantics, originated from Pearce's equilibrium logic [4].

The *reduct* of a rule R relative to a closure C, denoted by R^C, is the rule obtained from R by simultaneously replacing every maximal subrule not satisfied by C with $\bot$. A closure C is said to be a *candidate belief*[4] of a rule R if it is the minimal closure (in the sense of set inclusion) satisfying R^C. That is, $C \models_B R^C$ and there does not exist another closure $C_1 \subset C$ such that $C_1 \models_B R^C$. We say that two rules are *equivalent* if they have the same set of candidate beliefs. Clearly, this definition can be generalized to rule bases, similar for definitions presented later.

Example 3 (Example 2 continued). Consider the example $\sim F_1 \Rightarrow F_2$ again. Assume that F_1 and F_2 are not related in the underlying logic[5]. Let C_1 be the closure $Cn(\{F_1\})$ and C_2 be the closure $Cn(\{F_2\})$. Then, $(\sim F_1 \Rightarrow F_2)^{C_1}$ is $\bot \Rightarrow F_2$. Of course, C_1 is a model of $\bot \Rightarrow F_2$. However, $Cn(\emptyset)$ is also a model of $\bot \Rightarrow F_2$. Thus, C_1 is not a candidate belief of $\sim F_1 \Rightarrow F_2$. On the other hand, C_2 is the minimal closure satisfying $(\sim F_1 \Rightarrow F_2)^{C_2}$, which is $\sim\bot \Rightarrow F_2$. Thus, C_2 is a candidate belief of $\sim F_1 \Rightarrow F_2$.

We introduce a notion of *strong equivalence* between two rules. The notion of strong equivalence, introduced by Lifschitz [7] for answer set programming, and extended to default logic by Turner [8], plays a very important role in default reasoning. Two rules R_1 and R_2 are said to be *strongly equivalent* iff for all other rules R_3, $R_1 \,\&\, R_3$ has the same set of candidate beliefs as $R_2 \,\&\, R_3$.

[4] This is different from the notion of belief or knowledge in epistemic reasoning.

[5] There are four possible relationships between F_1 and F_2: (a) there is no closures containing both F_1 and F_2; (b) $\{F_1\} \models_{\mathcal{L}} F_2$; (c) $\{F_2\} \models_{\mathcal{L}} F_1$, or (d) none of the above. In the first three cases, F_1 and F_2 are related.

We also define the equilibrium semantics for default reasoning. A bi-level interpretation $\langle C_1, C_2\rangle$ is said to be an *equilibrium model* of a rule R iff (a) $\langle C_1, C_2\rangle$ is a bi-level model of R; (b) $C_1 = C_2$; (c) there does not exist $C_1' \subset C_1$ such that $\langle C_1', C_2\rangle$ is also a bi-level model of R.

Equilibrium semantics is essentially a fixed point semantics. In this sense, it shares the same basic idea of the extension semantics. Both of them can be viewed as three steps. First, guess a possible set of information. Second, derive a minimal set of information by fixing the guess. Finally, if these two sets coincide with each other, then it is a possible candidate belief.

Example 4 (Example 3 continued). Again, consider the example $\sim F_1 \Rightarrow F_2$. Let C_0, C_1 and C_2 be three closures $Cn(\emptyset)$, $Cn(\{F_1\})$ and $Cn(\{F_2\})$ respectively. We have that $\langle C_1, C_1\rangle$ is a bi-level model of $\sim F_1 \Rightarrow F_2$, so is $\langle C_0, C_1\rangle$. Thus, $\langle C_1, C_1\rangle$ is not an equilibrium model of $\sim F_1 \Rightarrow F_2$. On the other hand, $\langle C_2, C_2\rangle$ is a bi-level model of $\sim F_1 \Rightarrow F_2$, and there is no other closure C' such that $C' \subset C_2$ and $\langle C', C_2\rangle$ is also a bi-level model of $\sim F_1 \Rightarrow F_2$. Thus, $\langle C_2, C_2\rangle$ is an equilibrium model of $\sim F_1 \Rightarrow F_2$.

Certainly, the four semantics, basic semantics, bi-level semantics, extension semantics and equilibrium semantics are closely related.

Proposition 8. *Let Δ be a rule base and C a closure. If C is a candidate belief of Δ, then $C \models_B \Delta$.*

However, the converse of Proposition 8 does not hold in general. For instance, $Cn(\{F_1\})$ is a model of $\sim F_1 \Rightarrow F_2$ but not a candidate belief of it.

Proposition 9. *Let Δ be a rule base and C_1 and C_2 two closures. $\langle C_1, C_2\rangle \models_{BI} \Delta$ iff $C_1 \models_B \Delta^{C_2}$.*

Theorem 2. *Let Δ be a rule base and C a closure. C is a candidate belief of Δ iff $\langle C, C\rangle$ is an equilibrium model of Δ.*

Proposition 10. *Let R_1 and R_2 be two rules. R_1 and R_2 are strongly equivalent iff $\models_{BI} R_1 \Leftrightarrow R_2$.*

Corollary 2. *Let Δ be a rule base and R a rule. $\Delta \models_{BI} R$ iff $\Delta \cup \{R\}$ is strongly equivalent to Δ.*

To sum up, we have defined three levels of semantics, the basic semantics lies on the underlying level and the bi-level semantics lies on the meta level, whilst on the middle level are two equivalent semantics, namely the extension semantics and the equilibrium semantics. Although basic semantics, equilibrium semantics and reduction-style semantics are used for answer set programming, the idea that they can be used in a much more general sense has not been proposed yet. In addition, the idea of using bi-level semantics for meta level reasoning is a novel approach. The equivalence relations of those three levels are captured by weak equivalence, strong equivalence and equivalence respectively.

Proposition 11. *Let R_1 and R_2 be two rules. If R_1 and R_2 are strongly equivalent, then R_1 and R_2 are equivalent and weakly equivalent as well.*

Example 5 (Example 4 continued). Consider four rules $\sim F_1 \Rightarrow F_2$, F_2, $\sim\sim F_1 \wr F_2$ and $F_1 \wr F_2$ and assume that F_1 and F_2 are not related in the underlying logic (See Footnote 5). We have that $\sim F_2 \Rightarrow F_1$ is strongly equivalent to $\sim\sim F_1 \wr F_2$. $\sim F_1 \Rightarrow F_2$ is equivalent to F_2, but neither strongly equivalent nor weakly equivalent to F_2. $\sim F_1 \Rightarrow F_2$ is weakly equivalent to $F_1 \wr F_2$, but neither strongly equivalent nor equivalent to $F_1 \wr F_2$.

The results proposed in this section are not surprising since they hold for answer set programming as well. However, their proofs in general do not follow directly from the simplest cases since the underlying logic is not simply a set. There are many features which can be exploited in set inclusion. For instance, in set inclusion, a set of atoms is a closure. However, it might be not the case for an arbitrary logic. In addition, the atoms in set are not related, but they may have very complex relationships in an arbitrary logic.

4 The Underlying Logic

In this section, we show that the logic framework presented above is powerful enough to capture various existing approaches by applying the underlying logic to different classes. Due to a space limit, we only briefly outline the basic ideas and results in this paper and leave backgrounds, detailed comparisons and discussions to a future full version.

4.1 Set Inclusion (with Classical Negation)

As we have shown in Section 2, set inclusion can be considered as a logic. Similarly, set inclusion with classical negation can be treated as a logic as well. Let $Atom$ be a set of atoms and Lit the set of literals, i.e., atoms or their classical negations. The formulas in the language $\mathcal{S}^{\neg}$ of set inclusion with classical negation are defined as elements in Lit. Let Γ be a set of formulas (i.e. a set of literals) and F a formula (i.e. a literal). $\Gamma \models_{\mathcal{S}^{\neg}} F$ iff (a) $F \in \Gamma$ or (b) there exists an atom a such that $a, \neg a \in \Gamma$. Clearly, this entailment relation satisfies reflexivity and transitivity.

Answer set programming (with classical negation) corresponds to default reasoning in meta level language when the underlying logic is set inclusion (with classical negation). On the other hand, it also explains why the answer set semantics for logic programs with classical negation work very well.

Theorem 3. *Let P be a disjunctive logic program (with classical negation) [9] and X a set of atoms (literals). X is an answer set of P iff X is a candidate belief of $\widehat{P}$ in $\mathcal{ML}(\mathcal{S})$ ($\mathcal{ML}(\mathcal{S}^{\neg})$), where $\widehat{P}$ is the set of meta level rules obtained from P by replacing each rule $p_1 \mid \ldots \mid p_n \leftarrow q_1, \ldots, q_m, \mathtt{not}\, r_1, \ldots, \mathtt{not}\, r_l$ by a meta level rule $q_1 \,\&\, \ldots \,\&\, q_m \,\&\, \sim r_1 \,\&\, \ldots \,\&\, \sim r_l \Rightarrow p_1 \wr \ldots \wr p_n$.*

Clearly, Theorem 3 also holds for normal logic programming [10]. More generally, it holds for Ferraris' answer set semantics for propositional theories [6] as well.

Theorem 4. *Let Γ be a set of propositional formulas and X a set of atoms. X is an answer set of Γ iff X is a candidate belief of $\widehat{\Gamma}$ in $\mathcal{ML}(\mathcal{S})$, where $\widehat{\Gamma}$ is the set of meta level rules obtained from Γ by replacing every classical propositional connectives with corresponding rule connectives.*

4.2 Propositional Logic

Certainly, classical propositional logic $\mathcal{CL}$ is a typical example of the underlying logic. The following theorem shows that Zhou et al.'s general default logic is a special case of the logic framework by applying the underlying logic to propositional logic.

Theorem 5. *Let Δ be a rule base in general default logic [3]. A theory T is an extension of Δ iff T is a candidate belief of Δ in $\mathcal{ML}(\mathcal{CL})$.*

As shown in [3], Reiter's default logic in the propositional case [11] and Gelfond et al.'s disjunctive default logic [12] are special cases of general default logic. Therefore these two approaches are also special cases of default reasoning of $\mathcal{ML}(\mathcal{CL})$.

4.3 First Order Logic: Closed Case

Let $\mathcal{FOL}$ be a first order language. By $\mathcal{FOL}_{\mathcal{S}}$, we denote the subclass of $\mathcal{FOL}$ by restricting the formulas to sentences, i.e., first order formulas without free variables.

Theorem 6. *A set of sentences is an extension of a closed default theory $\langle D, W\rangle$ iff it is a candidate belief of $W \cup \widehat{D}$ in $\mathcal{ML}(\mathcal{FOL}_{\mathcal{S}})$*[6].

4.4 Multi-context Logic

It is well argued that the notion of context plays a very important role in AI [13,14,15,16]. A context of an agent about the environment represents its own (local) subjective view of the environment. There is an increasing interest on formalizing a multi-context language, which defines not only the information of a number of contexts themselves but also the information of interrelationships among them.

Given a set $\mathcal{L}_1, \ldots, \mathcal{L}_n$ of n languages and their corresponding entailment relations satisfying both reflexivity and transitivity, we define a multi-context language $\mathcal{L}_1 \times \ldots \times \mathcal{L}_n$. The formulas in $\mathcal{L}_1 \times \ldots \times \mathcal{L}_n$ are labeled formulas, which have the form $\langle k, F\rangle$, where k is a label to denote which context it comes from and F is a formula in $\mathcal{L}_k$. A formula $\langle k, F\rangle$ is entailed by a set Γ of formulas in $\mathcal{L}_1 \times \ldots \times \mathcal{L}_n$ iff F is entailed in the logic $\mathcal{L}_k$ by the set of formulas in Γ labeled by k. Formally,

$$\Gamma \models_{\mathcal{L}_1 \times \ldots \times \mathcal{L}_n} \langle k, F\rangle \text{ iff } \{G \mid \langle k, G\rangle \in \Gamma\} \models_{\mathcal{L}_k} F.$$

Clearly, $\models_{\mathcal{L}_1 \times \ldots \times \mathcal{L}_n}$ satisfies reflexivity and transitivity as well.

A set Γ of formulas in $\mathcal{L}_1 \times \ldots \times \mathcal{L}_n$ can be equivalently written as an n-tuple $(\Gamma_1, \ldots, \Gamma_n)$, where $\Gamma_k = \{G \mid \langle k, G\rangle \in \Gamma\}$. Under this reformulation, we show that Brewka et al.'s contextual default reasoning, which can be considered as a "syntactical" counterpart of Roelofsen and Serafini's information chain approach [15], is a special case of the framework when the underlying logic is a multi-context propositional logic.

[6] $\widehat{D}$ is the set of rules in $\mathcal{ML}(\mathcal{FOL}_{\mathcal{S}})$ by rewriting each rule $F_1, MF_2, \ldots, MF_n/F_{n+1}$ in D to $F_1 \,\&\, {\sim}\neg F_2 \,\&\, \ldots \,\&\, {\sim}\neg F_n \Rightarrow F_{n+1}$.

Theorem 7. *Let $\mathcal{L}_1, \ldots, \mathcal{L}_n$ be n propositional languages (built over different sets of atoms). $(\Gamma_1, \ldots, \Gamma_n)$ is a contextual extension of a normal multi-context system C [16] iff $(\Gamma_1, \ldots, \Gamma_n)$ is a candidate belief of $\widehat{C}$ in $\mathcal{ML}(\mathcal{L}_1 \times \ldots \times \mathcal{L}_n)$*[7].

Furthermore, Brewka et al.'s approach is a homogeneous one (i.e. all the contexts are propositional languages), whilst our approach allows heterogenous contexts. Our approach is also an generalization of Giunchiglia's heterogenous multi-context logic [14], which does not consider nonmonotonic rules.

A related work is due to Brewka and Eiter [2]. They also intend to integrate heterogenous contexts by nonmonotonic rules. However, in their approach, the underlying logic is defined as a tuple (KB, BS, ACC), where KB is the set of all possible knowledge bases, BS is the set of all possible belief sets, and ACC is a function from KB to 2^{BS}, for describing the "semantics" of this logic. A logic in our sense can be decried as a special kind of this tuple as follows: KB are the sets of formulas in $\mathcal{L}$; BS are the sets of closures; ACC is the closure operator. When restricting Brewka and Eiter's sense of logic to ours, their approach of equilibria coincides with default reasoning of the meta level language by applying the underlying logic to multi-context logic.

5 Reducing into Answer Set Programming

In this section, we show that the logic framework proposed in this paper can be reduced into its simplest case, namely answer set programming, by identifying the internal relationships among formulas in the underlying logic. Here, we consider the general case of answer set programming in the propositional case [6].

Let R be a rule. We write $Fact(R)$ to denote the set of subfacts of R. Suppose that $Fact(R) = \{F_1, \ldots, F_n\}$. We introduce n new atoms $P = \{p_1, \ldots, p_n\}$ associated with each fact in $Fact(R)$. By $P(R)$ we denote the answer set program obtained from R by simultaneously replacing each occurrence of $F_i, (1 \leq i \leq n)$ in R with p_i. By $I(R)$ we denote the programs of all rules of the following form:

$$p_{i_1} \,\&\, p_{i_2} \,\&\, \ldots \,\&\, p_{i_k} \Rightarrow p_j,$$

where $\{i_1, i_2, \ldots, i_k, j\} \subseteq \{1, \ldots, n\}$ such that $\{F_{i_1}, F_{i_2}, \ldots, F_{i_k}\} \models_{\mathcal{L}} F_j$. By $Tr(R)$ we denote the program $\{P(R), I(R)\}$.

Let Γ and Γ' be two sets of formulas in $\mathcal{L}$ such that $\Gamma \subseteq \Gamma'$. We say that Γ is *maximal* to Γ' iff for all formulas $F \in \Gamma' \backslash \Gamma$, $\Gamma \not\models_{\mathcal{L}} F$.

Proposition 12. *Let R be a rule and C a closure. If C is a candidate belief of R, then there exists a subset Γ of $Fact(R)$ such that $C = Cn(\Gamma)$.*

Theorem 8. *Let R be a rule and $Fact(R) = \{F_1, \ldots, F_n\}$. Let $P = \{p_1, \ldots, p_n\}$ be n new atoms associated with each fact in $Fact(R)$.*

[7] Here, $\widehat{C}$ is the set of rules in $\mathcal{ML}(\mathcal{L}_1 \times \ldots \times \mathcal{L}_n)$ obtained from C by rewriting each fact F in W_i to $\langle i, F_i \rangle$, and each rule $\langle c_1, G_1 \rangle, \ldots, \langle c_m, G_m \rangle : \langle c_{m+1}, H_1 \rangle, \ldots, \langle c_{m+n}, H_n \rangle / F$ in D_i to $\langle c_1, G_1 \rangle \,\&\, \ldots \,\&\, \langle c_m, G_m \rangle \,\&\, {\sim}\langle c_{m+1}, \neg H_1 \rangle \,\&\, \ldots \,\&\, {\sim}\langle c_{m+n}, \neg H_n \rangle \Rightarrow \langle i : F \rangle$, where F, G_i, H_j are propositional formulas in corresponding contexts.

1. *If* $\{p_{i1}, \ldots, p_{ik}\}$ *is an answer set of* $Tr(R)$*, then* $Cn\{F_{i1}, \ldots, F_{ik}\}$ *is a candidate belief of* R *and* $\{F_{i1}, \ldots, F_{ik}\}$ *is maximal to* $Fact(R)$*.*
2. *If* C *is a candidate belief of* R *and* $\{F_{i1}, \ldots, F_{ik}\}$ *is the set of formulas obtained in Proposition 12 such that* $C = Cn(\{F_{i1}, \ldots, F_{ik}\})$*, then* $\{p_{i1}, \ldots, p_{ik}\}$ *is an answer set of* $Tr(R)$*.*

Theorem 9. *Let* R *be a rule.* $\models_{BI} R$ *iff all HT-models of* $I(R)$ *are also HT-models of* $P(R)$*.*

Intuitively, the rule R is constructed from $Fact(R)$ by rule connectives. $P(R)$ represents the structure of R (i.e. the way of constructing R), while $I(R)$ identifies all the internal relationships among $Fact(R)$. Theorem 8 and 9 show that, both meta level reasoning and default reasoning in any meta level language can be captured in answer set programming by separating the structure of rules and the interrelationships among underlying facts.

The translation introduces n new atoms, where n is the number of facts in $Fact(R)$. Clearly, n is polynomial, in fact linear, in the length of R. Interestingly, the original atoms in R no longer occur in $Tr(R)$. However, in some cases, n could be exponential in the number of the original atoms since atoms can compose exponential number of formulas in some underlying logics (e.g. propositional logic).

One of the most important problems is whether this translation is polynomial or not. Unfortunately, although $P(R)$ is linear in the size of R, $I(R)$ may contain exponential number of rules. The reason is that there may be exponential number of such internal relationships among $Fact(R)$. Observing that not all rules in $I(R)$ are necessary, we can pick up those "minimal" ones, namely, the set of premises is the minimal set satisfying the consequence in $Fact(R)$. However, even only taken these internal relationships into account, the number is still exponential. For example, in classical propositional logic, let $Atom = \{a_1, \ldots, a_m\}$ be m atoms. Let $F_{ij}, (1 \leq i \leq m), (1 \leq j \leq m)$ be the formula $a_i \rightarrow a_j$. Then, we have m^2 formulas. However, we have exponential number of such internal relationships. For instance, for any set of atoms $a_{i1}, \ldots, a_{ik}$ different from two atoms a_i, a_j, $\{a_i \rightarrow a_{i1}, a_{i1} \rightarrow a_{i2}, \ldots, a_{ik-1} \rightarrow a_{ik}, a_{ik} \rightarrow a_j\}$ is a minimal set satisfying $a_i \rightarrow a_j$.

Despite this negative result, the translation is not only of theoretical interests but also of practical uses. It enables us to easily analyze the meta level language in small-scale case studies. Also, it is a useful tool to investigate properties in meta level language. For instance, the following propositions follow from Theorem 8 and 9 straightforwardly.

Corollary 3. *If* $\mathcal{L}$ *is a decidable language, then both meta level reasoning and default reasoning of* $\mathcal{ML}(\mathcal{L})$ *are decidable.*

Corollary 4. *Contextual ASP [16] has the same computational complexity as normal logic programming.*

6 Conclusion

In this paper, we proposed a logic framework for meta level reasoning as well as default reasoning about meta level rules. In this framework, meta level reasoning is the

reasoning task of how to deduce new meta level rules by giving a set of rules, while default reasoning is the problem of what are the possible candidate beliefs by giving them. Default reasoning has attracted a lot of attentions in the past three decades [3,6,8,9,10,11,12,16,17]. On the other hand, meta level reasoning, although also important, was relatively less studied. A closely related topic is normative reasoning [18], which can be considered as a fragment of meta level reasoning of default logic. Another topic is so-called SE-consequence [19,20] in answer set programming, which actually coincides with meta level reasoning task of answer set programming.

Some technical results, although not trivial, might not be surprising since they hold for corresponding cases of answer set programming and default logic as well [3, 5]. However, surprisingly, all these can be summarized in a general framework with simple semantics. Hence, we argue that, this framework, indeed, captures the nature of both meta level reasoning and default reasoning in a general sense.

It is worth mentioning that default reasoning is nonmonotonic (in the sense of skeptical reasoning or credulous reasoning), whilst meta level reasoning is actually monotonic (See Proposition 5). Furthermore, meta level reasoning also satisfies reflexivity and transitivity. This means that meta level reasoning itself can also be treated as a logic in our sense. However, default reasoning is not. In other words, in this framework, "default" is not a logic but a meta level reasoning task.

We demonstrated this framework's expressiveness to capture several existing approaches of default reasoning by applying the underlying logic to different classes. More precisely, answer set programming, default logic in propositional case, default logic in closed first order case and contextual default logic coincide with default reasoning of meta level language of set inclusion, propositional logic, first order logic and multi-context logic respectively (See Theorem 3-7).

Also, the framework will initiate some new promising formalisms. One of them is to consider description logic, for instance $\mathcal{SHIQ}$, as the underlying logic. This provides a natural combination of description logic and rule-based formalism, which is a crucial step to fulfil the blueprint of Semantic Web Initiative [21,22]. However, rule connectives considered in this paper are basically propositional. In other words, meta level rules with free variables cannot be represented in this approach. One possible way to overcome this barrier is to use the technique of grounding [9,10]. That is, to define a first order meta level language powerful enough to represent rules with variables, and then to transfer them to propositional meta level rules by grounding for all instances. However, this topic is beyond the scope of this paper. We leave it to our future investigations. Certainly, there are many other interesting and important candidates of the underlying logic, such as epistemic logic, logic of multi-agents, temporal logics, logics of uncertainty, and so on. Many of them are worth pursuing. We leave them to our future work as well.

We showed that both meta level reasoning and default reasoning in a general sense can be reduced to its simplest case (Theorem 8 and 9), namely answer set programming, by identifying the internal relationships (represented by $I(R)$ in the translation) among formulas in the underlying logic. This provides a powerful tool to study the general framework without going through the details of the underlying logic.

To sum up, the main contributions of this framework are as follows. Firstly, it unifies a bunch of existing approaches for default reasoning, and it can be easily seen that this

approach can also initiate other promising paradigms of default reasoning. In addition, it suggests a new reasoning task, namely meta level reasoning, for deriving new meta level rules given a rule base. Finally, it can be interestingly reduced to the simplest case, namely answer set programming.

References

1. Szabo, M.: The collected papers of Gerhard Gentzen (1969)
2. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI, pp. 385–390 (2007)
3. Zhou, Y., Lin, F., Zhang, Y.: General default logic. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483, pp. 241–253. Springer, Heidelberg (2007)
4. Pearce, D.: A new logical characterisation of stable models and answer sets. In: NMELP, pp. 57–70 (1997)
5. Ferraris, P., Lifschitz, V.: Mathematical foundations of answer set programming. In: We Will Show Them! (1), pp. 615–664 (2005)
6. Ferraris, P.: Answer sets for propositional theories. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 119–131. Springer, Heidelberg (2005)
7. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic 2, 526–541 (2001)
8. Turner, H.: Strong equivalence for logic programs and default theories (made easy). In: Eiter, T., Faber, W., Truszczyński, M. (eds.) LPNMR 2001. LNCS (LNAI), vol. 2173, pp. 81–92. Springer, Heidelberg (2001)
9. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385 (1991)
10. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP, pp. 1070–1080 (1988)
11. Reiter, R.: A logic for default reasoning. Artificial Intelligence 13, 81–132 (1980)
12. Gelfond, M., Lifschitz, V., Przymusinska, H., Truszczyński, M.: Disjunctive defaults. In: KR, pp. 230–237 (1991)
13. McCarthy, J.: Notes on formalizing context. In: IJCAI, pp. 555–560 (1993)
14. Giunchiglia, F.: Contextual reasoning. Epistemologia, 345–364 (1993)
15. Roelofsen, F., Serafini, L.: Minimal and absent information in contexts. In: IJCAI, pp. 558–563 (2005)
16. Brewka, G., Roelofsen, F., Serafini, L.: Contextual default reasoning. In: IJCAI, pp. 268–273 (2007)
17. Lin, F., Shoham, Y.: A logic of knowledge and justified assumptions. Artificial Intelligence 57, 271–289 (1992)
18. Geffner, H., Pearl, J.: Conditional entailment: bridging two approaches to default reasoning. Artificial Intelligence 53(2-3), 209–244 (1992)
19. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Simplifying logic programs under uniform and strong equivalence. In: Lifschitz, V., Niemelä, I. (eds.) LPNMR 2004. LNCS (LNAI), vol. 2923, pp. 87–99. Springer, Heidelberg (2004)
20. Wong, K.: Sound and complete inference rules for se-consequence. JAIR 31, 205–216 (2008)
21. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. In: KR, pp. 141–151 (2004)
22. Motik, B., Rosati, R.: A faithful integration of description logics with logic programming. In: IJCAI, pp. 477–482 (2007)

Rule Calculus: Semantics, Axioms and Applications

Yi Zhou and Yan Zhang

Intelligent Systems Lab
School of Computing and Mathematics
University of Western Sydney
Locked Bag 1797, NSW, Australia

Abstract. We consider the problem of how a default rule can be deduced from a default theory. For this purpose, we propose an axiom system which precisely captures the deductive reasoning about default rules. We show that our axiomatic system is sound and complete under the semantics of the logic of here-and-there. We also study other important properties such as substitution and monotonicity of our system and prove the essential decision problem complexity. Finally, we discuss applications of our default rule calculus to various problems.

1 Introduction

Default logic is one of the predominant approaches for nonmonotonic reasoning. Many research topics related to default logic have been considerably studied including extensions, variations and alternatives [1, 2, 3] of Reiter's original definition [4], computational issues [5, 6] and so on.

However, one problem in default logic has been neglected in previous research. That is, how can we deduce a default rule from a default theory? In other words, in which sense can we say that a default rule is a consequence of a given default theory? This problem of rule deduction is of special interests from both theoretical and practical viewpoints. For instance, we may consider whether we can have a deductive system to formalize reasoning about default rules, and also implement a nonmonotonic knowledge system for more complex decision making where a decision could be a default rule. Quite obviously, to achieve such goals, the first fundamental task is that we should develop a logic or calculus for default rule reasoning.

In this paper, we propose a logical calculus, called *default rule calculus* (rule calculus for short), to address the problem of rule deduction. We first extend the logic of here-and-there to define a model-theoretical semantics for rule calculus, and discuss its relationships to the extension semantics. Then we define an axiom system, which extends both classical propositional calculus and the intermediate logic G3 (Gödel's 3-valued logic) [7], and prove its soundness and completeness. We further investigate some important properties of our system such as substitution and monotonicity, and prove the essential decision problem complexity. Finally, we discuss how our work can be applied to various problems such as the extension of generality among default rules and revision of nonmonotonic knowledge bases.

The reasons why we use the logic of here-and-there are as follows. Firstly, it is well studied in philosophical logic and it also has a simple axiomatic counterpart, namely

S. Hölldobler, C. Lutz, and H. Wansing (Eds.): JELIA 2008, LNAI 5293, pp. 416–428, 2008.

G3 [7]. Secondly, it is proven to be a very useful foundation of answer set programming [2, 8, 9, 10]. As pointed out in [3], answer set programming is a special case of default logic by restricting the propositional formulas to atoms. Thus, the extended version of the logic of here-and-there should, analogously, serve as a foundation of default logic. Thirdly, the logic of here-and-there naturally captures [2, 11] the notion of strong equivalence [11], which is argued to be the notion of "real equivalence" among answer set programs. As a consequence, it will also capture real equivalence among default rules.

In this paper, we will use general default logic [3] as a basis for the development of default rule calculus. Reasons for this are of three aspects: firstly, general default logic is a generalization of Reiter's default logic [4], Gelfond et al.'s disjunctive default logic [1] and Turner's nested default logic [2], which provides the most generalized default reasoning in the default logics paradigm; secondly, the syntax of general default logic is defined as arbitrary compositions of propositional formulas and rule connectives; finally, its extension semantics is defined in a very simple way as that for answer set programming [9].

The rest of the paper is organized as follows. In Section 2, we briefly review the syntax and semantics of general default logic, and then define the semantics of rule calculus. In Section 3, we present an axiom system for rule calculus and prove its soundness and completeness result. We then study relevant important properties of our axiom system for default rule calculus in Section 4. In Section 5, we discuss possible applications of rule calculus. Finally, in Section 6 we conclude the paper with some remarks.

2 Rule Calculus: Syntax and Semantics

To begin with, we recall some basic notions of classical propositional logic. The classical propositional language $\mathcal{L}$ is defined recursively by a set $Atom$ of atoms (or primitive propositions, variables) and a set of classical connectives $\bot$, $\rightarrow$ and $\neg$. Other connectives, such as $\top$, $\wedge$, $\vee$, $\leftrightarrow$, are defined as usual. Literals are atoms and their negations. The satisfaction relation $\models$ is defined as usual. A set of formulas in $\mathcal{L}$ is said to be a *theory* iff it is closed under classical entailment. Moreover, it is inconsistent iff it contains both a formula F and $\neg F$, otherwise, it is consistent. Let Γ be a set of formulas, by $Th(\Gamma)$ we denote the theory containing all formulas entailed by Γ. For convenience, we also use a set of formulas Γ to denote a theory T if $T = Th(\Gamma)$.

The language $\mathcal{R}$ of general default logic [3] is defined upon $\mathcal{L}$ by adding a set of rule connectives $\Rightarrow$, $\&$ and $\mid$ recursively:

$$R ::= F \mid R \Rightarrow R \mid R \,\&\, R \mid R \mid R,$$

where $F \in \mathcal{L}$. $-R$ and $R_1 \Leftrightarrow R_2$ are considered as shorthand of $R \Rightarrow \bot$ and $(R_1 \Rightarrow R_2)\ \&\ (R_2 \Rightarrow R_1)$ respectively. The order of priority for these connectives are

$$\{\neg\} > \{\wedge, \vee\} > \{\rightarrow, \leftrightarrow\} > \{-\} > \{\ \&\ ,\ \mid\ \} > \{\Rightarrow, \Leftrightarrow\}.$$

Formulas in $\mathcal{R}$ are called *rules*, whilst formulas in $\mathcal{L}$ are called *facts*. A *rule base* is a set of rules. The satisfaction relation $\models$ between a theory T and a rule R is defined recursively as follows:

- If R is a fact, then $T \models R$ iff $R \in T$.
- $T \models R \,\&\, S$ iff $T \models R$ and $T \models S$;
- $T \models R \mid S$ iff $T \models R$ or $T \models S$;
- $T \models R \Rightarrow S$ iff $T \not\models R$ or $T \models S$.

Hence, if T is consistent, then $T \models -R$ iff $T \not\models R$. If T is inconsistent, then for every rule R, $T \models R$. We say that T is a *model* of R iff $T \models R$.

The extension semantics of general default logic defined in [3] is not defined as the same as Reiter's original definition [4]. However, it is defined in a reduction-style similarly to that of answer set programming [9]. The *reduct* of a rule R relative to a theory T, denoted by R^T, is the rule obtained from R by replacing every maximal subrule[1] of R which is not satisfied by T with $\bot$. The reduct of a rule base relative to a theory is defined as the set of reducts of its rules relative to this theory. A theory T is said to be an *extension* of a rule base Δ iff it is the minimal (in the sense of set inclusion) theory satisfying Δ^T.

As shown in [3], Reiter's default logic [4] in propositional case is a special case of general default logic by restricting the rules to the following form

$$F \,\&\, -G_1 \,\&\, \ldots \,\&\, -G_n \Rightarrow H,$$

where $n \geq 0$, F, $G_i, (1 \leq i \leq n)$ and H are facts. Yet, under the context of Reiter's default logic, this form is represented as

$$\frac{F : M(\neg G_1), \ldots, M(\neg G_n)}{H}.$$

Similarly, both Gelfond et al.'s disjunctive default logic [1] and Turner's nested default logic [2] are also special cases of general default logic.

Here, we adopt Turner's (Section 7 in [2]) extended notion of Heyting's logic of here-and-there, introduced by Pearce [10] into answer set programming, as the basic semantics for general default logic. An *HT-interpretation* is a pair $\langle T_1, T_2\rangle$, where T_1 and T_2 are theories such that $T_1 \subseteq T_2$. The satisfaction relation $\models$[2] between an HT-interpretation $\langle T_1, T_2\rangle$ and a rule R is defined recursively:

- for a fact F, $\langle T_1, T_2\rangle \models F$ iff $F \in T_1$;
- $\langle T_1, T_2\rangle \models R_1 \,\&\, R_2$ iff $\langle T_1, T_2\rangle \models R_1$ and $\langle T_1, T_2\rangle \models R_2$;
- $\langle T_1, T_2\rangle \models R_1 \mid R_2$ iff $\langle T_1, T_2\rangle \models R_1$ or $\langle T_1, T_2\rangle \models R_2$;
- $\langle T_1, T_2\rangle \models R_1 \Rightarrow R_2$ iff
  1. $\langle T_1, T_2\rangle \not\models R_1$ or $\langle T_1, T_2\rangle \models R_2$, and
  2. $T_2 \models R_1 \Rightarrow R_2$.

We say that $\langle T_1, T_2\rangle$ is an *HT-model* of a rule R iff $\langle T_1, T_2\rangle \models R$. We say that a rule base Δ *implies* a rule R, denoted by $\Delta \models R$, iff all HT-models of Δ are also HT-models of R. It is easy to see that the HT-interpretation $\langle \bot, \bot\rangle$ is a model of all rules. We say

[1] The subrule relation is defined recursively: a) R_1 is a subrule of R_1, and b) R_1 and R_2 are subrules of $R_1 \,\&\, R_2$, $R_1 \mid R_2$ and $R_1 \Rightarrow R_2$.

[2] For convenience, we overload the notation $\models$ in this paper.

that a rule R is a *rule contradiction* iff $\langle \bot, \bot \rangle$ is the only HT-model of R. We say that a rule R is a *rule tautology* iff every HT-interpretation is an HT-model of R.

Intuitively, a theory T is a possible set of information of an agent about the world, and a rule R is a description or constraint about the information of the world. $T \models R$ means that the set T of information obeys (or does not violate) the constraint R; T is an extension of R means that T is one of the possible sets of information can be derived by giving the only constraint R. Given a rule base Δ and a rule R, $\Delta \models R$ means that the set of constraints Δ is more powerful than the constraint R. In other words, R can be eliminated by giving Δ.

Example 1. Consider the notorious bird-fly example. The statement "birds normally fly" can be represented as a default rule $bird \& - \neg fly \Rightarrow fly$. Given an instance of bird, represented as a fact $bird$, by the extension semantics, the only extension of the rule base $\{bird \ \& \ - \neg fly \Rightarrow fly, bird\}$ is $\{bird, fly\}$. However, fly is not a rule consequence of the rule base $\{bird \ \& \ - \neg fly \Rightarrow fly, bird\}$ since $\langle \{bird\}, \{bird, \neg fly\} \rangle$ is an HT-model of $\{bird \ \& \ - \neg fly \Rightarrow fly, bird\}$ but not an HT model of $\{bird, fly\}$. This shows a difference between the extension semantics and the HT-semantics.

Similarly, $bird \ \& \ - \neg fly \Rightarrow fly \not\models bird \Rightarrow fly$. However, one can check that all HT-models of $bird \Rightarrow fly$ are also HT-models of $bird \ \& \ - \neg fly \Rightarrow fly$. Therefore, $bird \Rightarrow fly \models bird \& - \neg fly \Rightarrow fly$. This means that, intuitively, the statement "birds normal fly" is strictly weaker than the statement "birds fly".

Example 2. Let p_1, p_2 and p_3 be three atoms. Consider the rule base $\{p_1 \ \& \ - \neg p_2 \Rightarrow p_2, p_2 \ \& \ - \neg p_3 \Rightarrow p_3\}$, which has an HT-model $\langle \{p_1\}, \{p_1, \neg p_2\} \rangle$. However, this HT-interpretation is not an HT-model of $p_1 \& - \neg p_3 \Rightarrow p_3$. This shows that $\{p_1 \& - \neg p_2 \Rightarrow p_2, p_2 \ \& \ - \neg p_3 \Rightarrow p_3\} \not\models p_1 \ \& \ - \neg p_3 \Rightarrow p_3$.

The extension semantics and HT-semantics of general default logic are closely related.

Proposition 1. *Let T_1 and T_2 be two consistent theories such that $T_1 \subseteq T_2$ and R a rule.*

- $T_1 \models R$ *iff* $\langle T_1, T_1 \rangle \models R$.
- *If* $\langle T_1, T_2 \rangle \models R$, *then* $T_2 \models R$.
- $\langle T_1, T_2 \rangle \models -R$ *iff* $T_2 \models -R$.

Proposition 2. *Let Δ be a rule base and F a fact. If $\Delta \models F$, then F is in all extensions of Δ.*

However, the converse of Proposition 2 does not hold in general. For instance, $\{p_1\}$ is the unique extension of $-p_2 \Rightarrow p_1$. Thus, p_1 is in all extensions of $-p_2 \Rightarrow p_1$. However, $-p_2 \Rightarrow p_1 \not\models p_1$ since $\langle \{p_2\}, \{p_2\} \rangle$ is an HT-model of $-p_2 \Rightarrow p_1$ but not an HT-model of p_1.

Proposition 3. *Let T_1 and T_2 be two theories such that $T_1 \subseteq T_2$ and Δ a rule base. $\langle T_1, T_2 \rangle$ is an HT-model of Δ iff $T_1 \models \Delta^{T_2}$.*

Proposition 4. *Let T be a theory and Δ a rule base. T is an extension of Δ iff $\langle T, T\rangle$ is an HT-model of Δ, and for all theories $T_1 \subset T$, $\langle T_1, T\rangle$ is not an HT-model of Δ.*

The notion of strong equivalence, introduced by [11] into answer set programming, plays an important role from both a theoretical and a practical viewpoints. A similar notion is introduced into default logic in [2]. We say that two rules R_1 and R_2 are *strongly equivalent*, denoted by $R_1 \equiv R_2$, iff for all other rules R_3, R_1 & R_3 has the same set of extensions as R_2 & R_3. Strong equivalence can also be defined in another way. That is, two rules R_1 and R_2 are strongly equivalent iff for all other rules R_3, R_3 has the same set of extensions as $R_3(R_1/R_2)$, where $R_3(R_1/R_2)$ is the rule obtained from R_3 by replacing every occurrence of R_1 in R_3 with R_2 simultaneously. It is clear that the notion of strong equivalence can be extended for the cases of rule bases.

In fact, strong equivalence in general default logic can be captured in the logic of here-and-there.

Proposition 5. *Let R_1 and R_2 be two rules. R_1 and R_2 are strongly equivalent iff they have the same set of HT-models in the logic of here-and-there. That is, $R_1 \equiv R_2$ iff $\models R_1 \Leftrightarrow R_2$.*

Since general default logic is both an extension of general logic programming and nested default logic, Proposition 5 is a generalization of both Proposition 2 in [9] and Theorem 3 in [2]. As a consequence of Proposition 5, checking whether a rule is implied by a rule base can be reduced to checking whether two rule bases are strongly equivalent.

Corollary 1. *Let Δ be a rule base and R a rule. $\Delta \models R$ iff $\Delta \cup \{R\}$ is strongly equivalent to Δ.*

Corollary 1 indicates the intuition behind rule deduction, that is, a rule R is a consequence of a rule base Δ means that R provides no more information by giving Δ. In other words, R can be eliminated by giving Δ.

3 Rule Calculus: Axiom System

In this section, we propose an axiom system for default rule calculus and prove the soundness and completeness results.

Axioms The axioms of rule calculus are:

A1. all tautologies in classical propositional logic.
A2. $(F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow F_2)$, where F_1 and F_2 are two facts.
A3. $R_1 \Rightarrow (R_2 \Rightarrow R_1)$.
A4. $(R_1 \Rightarrow (R_2 \Rightarrow R_3)) \Rightarrow ((R_1 \Rightarrow R_2) \Rightarrow (R_1 \Rightarrow R_3))$.
A5. $R_1 \Rightarrow (R_2 \Rightarrow (R_1 \,\&\, R_2))$.
A6. $R_1 \,\&\, R_2 \Rightarrow R_1$; $R_1 \,\&\, R_2 \Rightarrow R_2$.
A7. $R_1 \Rightarrow R_1 \mid R_2$; $R_2 \Rightarrow R_1 \mid R_2$.
A8. $(R_1 \Rightarrow R_3) \Rightarrow ((R_2 \Rightarrow R_3) \Rightarrow (R_1 \mid R_2 \Rightarrow R_3))$.
A9. $(R_1 \Rightarrow R_2) \Rightarrow ((R_1 \Rightarrow -R_2) \Rightarrow -R_1)$.
A10. $R_1 \mid (R_1 \Rightarrow R_2) \mid -R_2$.

Rules. The only[3] inference rule of rule calculus is

Rule Modus Ponens from R_1 and $R_1 \Rightarrow R_2$ to infer R_2.

Axiom 1 simply means that all classical tautologies are also rule tautologies; Axiom 2 is for bridging the gap between facts and rules; Axiom 3-9, together with Rule Modus Ponens, are generalization of the axiom system of intuitionistic logic [12]; Axiom A10 is the extended version of an additional axiom in the intermediate logic G3. That is, Axiom 3-10 and Rule Modus Ponens are generalization of the axiom system of G3 [10].

A rule R is said to be a *consequence* of a rule base Δ, denoted by $\Delta \vdash R$, iff there is a sequence of rules $R_1, \ldots, R_n$ such that $R_n = R$ and for each $i, (1 \leq i \leq n)$, either a) R_i is an instance of axiom, or b) R_i is in Δ, or c) R_i is obtained by an inference rule from some proceeding rules in this sequence. Such a sequence is called a *proof* (or *deduction*) of R from Δ. rules in Δ are called *premises*. A rule R is said to be a *rule theorem*, denoted by $\vdash R$, iff there exists a proof of R from the empty rule base. We use $\Delta \nvdash R$ to denote it is not the case that $\Delta \vdash R$.

As an example, we prove the following rule theorem.

Proposition 6. $\vdash (F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow -\neg F_2)$, *where* F_1 *and* F_2 *are two facts.*

Proof. We construct a proof as follows:

1. $(\neg F_2 \rightarrow \bot) \Rightarrow (\neg F_2 \Rightarrow \bot)$ by A2,
2. $(F_2 \Rightarrow -\neg F_2) \Rightarrow (F_1 \Rightarrow (F_2 \Rightarrow -\neg F_2))$ by A3,
3. $F_1 \Rightarrow (F_2 \Rightarrow -\neg F_2)$ by 1, 2 and RMP,
4. $(F_1 \Rightarrow (F_2 \Rightarrow -\neg F_2)) \Rightarrow ((F_1 \Rightarrow F_2) \Rightarrow (F_1 \Rightarrow -\neg F_2))$ by A4,
5. $(F_1 \Rightarrow F_2) \Rightarrow (F_1 \Rightarrow -\neg F_2)$ by 3, 4 and RMP,
6. $((F_1 \Rightarrow F_2) \Rightarrow ((F_1 \Rightarrow -\neg F_2)) \Rightarrow ((F_1 \rightarrow F_2) \Rightarrow ((F_1 \Rightarrow F_2) \Rightarrow (F_1 \Rightarrow -\neg F_2))))$ by A3,
7. $(F_1 \rightarrow F_2) \Rightarrow ((F_1 \Rightarrow F_2) \Rightarrow (F_1 \Rightarrow -\neg F_2))$ by 5, 6 and RMP,
8. $((F_1 \rightarrow F_2) \Rightarrow ((F_1 \Rightarrow F_2) \Rightarrow (F_1 \Rightarrow -\neg F_2))) \Rightarrow (((F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow F_2)) \Rightarrow ((F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow -\neg F_2)))$ by A4,
9. $((F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow F_2)) \Rightarrow ((F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow -\neg F_2))$ by 7, 8 and RMP,
10. $(F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow F_2)$ by A2,
11. $(F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow -\neg F_2)$ by 9, 10 and RMP.

This completes the proof.

A simple property following from the definition of proof of rule calculus is so-called compactness as follows.

Proposition 7 (Compactness). *Let* Δ *be a rule base and* R *a rule such that* $\Delta \vdash R$. *There exists a finite subset* Δ' *of* Δ *such that* $\Delta' \vdash R$.

Proposition 8 (Deduction theorem). *Let* Δ *be a rule base and* R_1 *and* R_2 *two rules.* $\Delta \cup \{R_1\} \vdash R_2$ *iff* $\Delta \vdash R_1 \Rightarrow R_2$.

[3] In fact, the Modus Ponens rule in classical logic is, of course, also an inference rule in rule calculus. However, since axiom A1 takes all classical tautologies into account, we can omit the classical Modus Ponens rule here.

Deduction theorem is a very useful tool for proving the consequence relationships in rule calculus. Consider the following example.

Example 3. [Example 1 continued] We prove that $bird \Rightarrow fly \vdash bird \& -\neg fly \Rightarrow fly$. By deduction theorem, we only need to prove $\{bird \Rightarrow fly, bird \ \& \ -\neg fly\} \vdash fly$. This is quite simple from A6 and RMP.

Proposition 9. $\vdash (R_1 \Rightarrow R_2) \Rightarrow (-R_2 \Rightarrow -R_1)$.

Proof. We construct a proof of $-R_1$ from $\{R_1 \Rightarrow R_2, -R_2\}$.

1. $-R_2 \Rightarrow (R_1 \Rightarrow -R_2)$ by A3,
2. $-R_2$ by premises,
3. $R_1 \Rightarrow -R_2$ by 1, 2 and RMP,
4. $(R_1 \Rightarrow R_2) \Rightarrow ((R_1 \Rightarrow -R_2) \Rightarrow -R_1)$ by A9,
5. $R_1 \Rightarrow R_2$ by premises,
6. $(R_1 \Rightarrow -R_2) \Rightarrow -R_1$ by 4, 5 and RMP,
7. $-R_1$ by 3, 6 and RMP.

Thus, $\{R_1 \Rightarrow R_2, -R_2\} \vdash -R_1$. By deduction theorem, $\{R_1 \Rightarrow R_2\} \vdash -R_2 \Rightarrow -R_1$. Again, by deduction theorem, $\vdash (R_1 \Rightarrow R_2) \Rightarrow (-R_2 \Rightarrow -R_1)$.

Proposition 10. *Let F, G and Q be three facts.*

1. $\vdash F \Rightarrow -\neg F$.
2. $\vdash (F \ \& \ G) \Leftrightarrow (F \wedge G)$.
3. $\vdash (-F \mid -G) \Leftrightarrow (\neg F \vee \neg G)$.
4. $\vdash (F \wedge Q \rightarrow G) \Rightarrow (F \ \& \ -G \Rightarrow -Q)$.
5. $\vdash (F \rightarrow G) \Rightarrow (-G \Rightarrow -F)$.

Theorem 1 (Soundness and completeness). *Let R be a rule. R is a rule tautology iff R is a rule theorem. That is, $\models R$ iff $\vdash R$.*

Proof. "soundness:" We first show that all instances of axioms are rule tautologies. As an example, we only present the proofs of A2 and A10 here. Let $\langle T_1, T_2\rangle$ be an HT-interpretation other than $\langle \perp, \perp\rangle$.

A2. Assume that $\langle T_1, T_2\rangle$ is not an HT-model of $(F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow F_2)$. Then, there are two cases. Case 1: $\langle T_1, T_2\rangle \models F_1 \rightarrow F_2$ and $\langle T_1, T_2\rangle \not\models F_1 \Rightarrow F_2$. That is, $T_1 \models F_1 \rightarrow F_2$ and a) $\langle T_1, T_2\rangle \models F_1$ and $\langle T_1, T_2\rangle \not\models F_2$ or b) $T_2 \not\models F_1 \Rightarrow F_2$. Thus, $T_1 \models F_1 \rightarrow F_2$ and a) $T_1 \models F_1$ and $T_1 \not\models F_2$ or b) $T_2 \models F_1$ and $T_2 \not\models F_2$. Whichever the case is, it leads to a contradiction. Case 2: $T_2 \not\models (F_1 \rightarrow F_2) \Rightarrow (F_1 \Rightarrow F_2)$. Then, $T_2 \models F_1 \rightarrow F_2$ and $T_2 \not\models F_1 \Rightarrow F_2$. That is, $T_2 \models F_1 \rightarrow F_2$ and $T_2 \models F_1$ and $T_2 \not\models F_2$, a contradiction.

A10. Assume that $\langle T_1, T_2\rangle$ is not an HT-model of $R_1 \mid (R_1 \Rightarrow R_2) \mid -R_2$. Then, $\langle T_1, T_2\rangle \not\models R_1$ and $\langle T_1, T_2\rangle \not\models R_1 \Rightarrow R_2$. Therefore $T_2 \not\models R_1 \Rightarrow R_2$. Thus, $T_2 \not\models R_2$. However, $\langle T_1, T_2\rangle \not\models -R_2$. Thus, By Proposition 1, $T_2 \not\models -R_2$, a contradiction.

We then show that all inferences rules preserve rule tautologies. Suppose that both R_1 and $R_1 \Rightarrow R_2$ are rule tautologies. Given an HT-interpretation $\langle T_1, T_2\rangle$, we have that $\langle T_1, T_2\rangle \models R_1$ and $\langle T_1, T_2\rangle \models R_1 \Rightarrow R_2$. Thus, $\langle T_1, T_2\rangle \models R_2$. This shows that R_2 is also a rule tautology. Hence, soundness holds.

"completeness:" As recently shown in [8], each formula in the logic of here-and-there is equivalent to a set of formulas of the following form:

$$p_1 \wedge \ldots \wedge p_n \wedge \neg p_{n+1} \wedge \ldots \wedge \neg p_m \rightarrow p_{m+1} \vee \ldots \vee p_k \vee \neg p_{k+1} \vee \ldots \vee \neg p_l,$$

where $p_i, (1 \leq i \leq l)$ are atoms. A similar result for rule calculus can be proved in the same way. That is, each rule is equivalent to a set of rules of the following form:

$$F_1 \,\&\, \ldots \,\&\, F_n \,\&\, -F_{n+1} \,\&\, \ldots \,\&\, -F_m \Rightarrow F_{m+1} \,|\, \ldots \,|\, F_k \,|\, -F_{k+1} \,|\, \ldots \,|\, -F_l, \quad (1)$$

where $F_i, (1 \leq i \leq l)$ are facts.

Thus, we only need to prove that for each rule R of form (1), if $\models R$, then $\vdash R$. For convenience, we assume that

$$R = F_1 \,\&\, \ldots \,\&\, F_n \,\&\, -G_1 \,\&\, \ldots \,\&\, -G_m \Rightarrow P_1 \,|\, \ldots \,|\, P_k \,|\, -Q_1 \,|\, \ldots \,|\, -Q_l.$$

Let $F = \bigwedge_{1 \leq i \leq n} F_i$ and $Q = \bigwedge_{1 \leq i \leq l} Q_i$ Then, one of the following statements must hold.

1. There exists $i, (1 \leq i \leq k)$ such that $F \rightarrow P_i$ is a classical tautology.
2. $F \rightarrow \neg Q$ is a classical tautology.
3. There exists $j, (1 \leq j \leq m)$ such that $(F \wedge Q) \rightarrow G_j$ is a classical tautology.

Suppose otherwise, then there exists a propositional assignment π_0 such that $\pi_0 \models F \wedge Q$; there exists a propositional assignment $\pi_i, (1 \leq i \leq k)$ such that $\pi_i \models F \wedge \neg P_i$; there exists a propositional assignment $\pi'_j, (1 \leq i \leq m)$ such that $\pi'_j \models F \wedge Q \wedge \neg G_j$. Let T_1 be the theory such that the set of its models is $\{\pi_0, \pi_i, \pi'_j, (1 \leq i \leq k), (1 \leq j \leq l)\}$, and T_2 be the theory such that the set of its models is $\{\pi'_j, (1 \leq j \leq l)\}$. It is easy to check that $\langle T_1, T_2\rangle$ is not an HT-model of R, a contradiction.

Thus, one of the three previous statements holds. Hence, R can be proved according to axioms and Proposition 6, Proposition 9 and Proposition 10. As an example, we prove the third case. Without loss of generality, suppose that $(F \wedge Q) \rightarrow G_1$ is a classical tautology. Then, by point 4 in Proposition 10, $F \,\&\, -G_1 \Rightarrow -Q$ is a rule tautology. Then, by point 2 and point 3 in Proposition 10, $F_1 \& \ldots \& F_n \& -G_1 \Rightarrow -Q_1 | \ldots | -Q_l$ is a rule tautology. By A6 and A7, R is a rule tautology.

From compactness, deduction theorem and soundness and completeness, we have the following result.

Corollary 2. *Let Δ be a rule base and R a rule. R is implied by Δ iff R is a consequence of Δ. That is, $\Delta \models R$ iff $\Delta \vdash R$.*

4 Other Properties

In this section, we discuss some other important properties of rule calculus.

From Proposition 10, one may claim that rule connectives and corresponding classical connectives play the same roles to some extent. However, this is not the case. For instance, $\nvdash (F|G) \Leftrightarrow (F \vee G)$, where F and G are two facts. As another example, from A2, $\vdash (F \rightarrow G) \Rightarrow (F \Rightarrow G)$. However, $\nvdash (F \Rightarrow G) \Rightarrow (F \rightarrow G)$. Consequently, $\nvdash -F \Rightarrow \neg F$ although $\vdash \neg F \Rightarrow -F$.

In fact, rule calculus is more like the intermediate logic G3. It is clear that the former is an extension of the latter. Hence, theorems not in G3 are not rule theorems in rule calculus. For example, $R \mid -R$ is not a rule theorem. On the other hand, theorems in G3 can be extended for rule calculus. For example, the following property holds.

Proposition 11. *Let R_1, R_2, R_3 and R_4 be four rules.*

1. $\{R_1 \Rightarrow R_2, R_2 \Rightarrow R_3\} \vdash R_1 \Rightarrow R_3$.
2. $\{R_1 \Rightarrow R_2, R_3 \Rightarrow R_4\} \vdash R_1 \mid R_3 \Rightarrow R_2 \mid R_4$.
3. $R_1 \Rightarrow - - R_1$.

Proposition 12. *Let R be a theorem in G3 composed from a set of atoms $P = \{p_1, \ldots, p_n\}$ and $\Delta = \{R_i, (1 \leq i \leq n)\}$ are n rules associated with each p_i. Then, $R(P/\Delta)$ is a rule theorem, where $R(P/\Delta)$ is the rule obtained from R by replacing every occurrence of $p_i, (1 \leq i \leq n)$ with corresponding R_i simultaneously.*

Proposition 13. *Let F_1 and F_2 be two facts. $F_1 \mid F_2 \vdash F_1 \vee F_2$.*

However, $F_1 \vee F_2 \nvdash F_1 \mid F_2$. For example, $\langle\{F_1 \vee F_2\}, \{F_1 \vee F_2\}\rangle$ is an HT-model of $F_1 \vee F_2$ but not an HT-model of $F_1 \mid F_2$.

Proposition 14 (Substitution). *Let R be a rule theorem and R_1 and R_2 two rules. $R(R_1/R_2)$, the rule obtained from R by replacing every occurrence of R_1 with R_2 simultaneously, is a rule theorem as well.*

Proposition 15. *Let F_1, F_2 and F_3 be three facts. $\{F_1 \Rightarrow F_2, F_2 \,\&\, - \neg F_3 \Rightarrow F_3\} \vdash F_1 \,\&\, - \neg F_3 \Rightarrow F_3$.*

However, $\{F_2 \Rightarrow F_3, F_1 \& - \neg F_2 \Rightarrow F_2\} \nvdash F_1 \& - \neg F_3 \Rightarrow F_3$ since $\langle\{p_1\}, \{p_1 \wedge \neg p_2\}\rangle$ is an HT-models of $\{p_2 \Rightarrow p_3, p_1 \& - \neg p_2 \Rightarrow p_2\}$ but not an HT-models of $p_1 \& - \neg p_3 \Rightarrow p_3$. In addition, as shown in Example 2, $\{F_1 \,\&\, - \neg F_2 \Rightarrow F_2, F_2 \,\&\, - \neg F_3 \Rightarrow F_3\} \nvdash F_1 \,\&\, - \neg F_3 \Rightarrow F_3$.

Interestingly, rule calculus is monotonic although the extension semantics of default logic is dealing with nonmonotonicity.

Proposition 16 (Monotonicity). *Let R be a rule and Δ and Δ' are two rule bases such that $\Delta \subseteq \Delta'$. If $\Delta \vdash R$, then $\Delta' \vdash R$.*

Theorem 2 (Complexity). *Checking whether a rule R has at least one HT-model is NP complete.*

Proof. Hardness is obvious since a fact is satisfiable iff it has at least one HT-model. For membership, we first prove a lemma by induction. Given two HT-interpretations $\langle T_1, T_2\rangle$ and $\langle T_1', T_2'\rangle$ and a set of facts Γ, if for all $F \in \Gamma$, $T_1 \models F$ iff $T_1' \models F$, and so

do T_2 and T_2', then for all rules R composed from Γ and rule connectives, $\langle T_1, T_2\rangle \models R$ iff $\langle T_1', T_2'\rangle \models R$.

Suppose that R is composed from the set of facts $\Gamma = \{F_1, \ldots, F_n\}$ and $\langle T_1, T_2\rangle \models R$. Without loss of generality, suppose that $T_1 \models F_i, (1 \leq i \leq m)$ and $T_1 \not\models F_i, (m < i \leq n)$. Therefore, there exists a propositional assignment $\pi_i, (m < i \leq n)$ such that $\pi_i \models \bigwedge_{1\leq j\leq m} F_j \wedge \neg F_i$. Let T_1' be the theory such that its models are $\pi_i, (m < i \leq n)$. It is easy to see that T_1 and T_1' agree the same on Γ. We can construct T_2' in the same way. We have that $T_1' \subseteq T_2'$. Therefore $\langle T_1', T_2'\rangle \models R$. This shows that if a rule R has a model, then it has a model which can be represented polynomially. It follows that checking whether a rule R has at least one HT-model is in NP.

It is well known that most of the decision problems in default logics lie on the second level of polynomial hierarchy [6], even restricted to some special subclasses [5]. Surprisingly, although rule calculus seems more complicated than others such as skeptical and credulous reasoning, its complexity is lower than them according to Theorem 2. This draws an opposite conclusion. That is, the problem of rule calculus is, indeed, simpler than other reasoning tasks of default logic. However, this does not mean that the former is weaker than the latter since they are dealing with different reasoning tasks of default rules.

5 Applications

Since the notion of rule deduction is an extension of deduction in propositional calculus, many propositional logic based deductive reasoning tasks can be lifted to corresponding cases in rule calculus. In this section, we briefly discuss three applications, which seem to be hard to deal with in default logic on their own. However, by using rule calculus, these problems can be easily solved. Due to a space limit, we only outline the basic ideas here.

Irrelevance in Default Logic

As pointed by Lang et al. [13], *irrelevance* is an important notion in propositional logic. According to Lang et al.'s definition, a propositional formula F is *irrelevant* to a set V of atoms iff there exists another formula G such that F is equivalent to G and $Atom(G) \cap V = \emptyset$, where $Atom(G)$ is the set of atoms appeared in G. The notion of irrelevance in rule calculus can be defined in a similar way. That is, a rule R_1 is *irrelevant* to a set V of atoms iff there exists another rule R_2 such that $\models R_1 \Leftrightarrow R_2$ and $Atom(R_2) \cap V = \emptyset$, where $Atom(R_2)$ is the set of atoms occurred in R_2.

Having defined the notion of irrelevance in rule calculus, we can define other related notions such as forgetting in a similar way as shown in [13].

Generality among Default Rules

The concept of generality is a foundational basis of inductive logic programming [14, 15] - a subfield of machine learning and has been successfully applied to some real domains. Inductive logic programming started from propositional logic [14] but then has focused on Horn clauses (namely logic programs) [15].

In propositional logic, the generality relationships between two formulas can be defined in a simple way as shown in [14]. That is, a formula F_1 is said to be *more general* than a formula F_2 iff $F_1 \models F_2$ and $F_2 \not\models F_1$. We can lift this notion to the case between two rules. A rule R_1 is said to be *more general* than a formula R_2 iff $R_1 \models R_2$ and $R_2 \not\models R_1$. Moreover, this notion can be easily extended to the cases with a background rule base. A rule R_1 is said to be *more general* than a rule R_2 *relative to* a rule base Δ iff $\Delta \cup \{R_1\} \models R_2$ and $\Delta \cup \{R_2\} \not\models R_1$. It is obvious that this definition is a generalization of the definition of generality in propositional calculus since all propositional formulas are also rules.

Inoue and Sakama [16] introduced several kinds of generality relationship between default theories. Although their definitions are based on disjunctive default logic, these can be easily extended to general default logic. We write $Ext(R)$ to denote the set of extensions of a rule R. Let R_1 and R_2 be two rules, according to Inoue and Sakama's definitions, R_1 is said to be *more* $\sharp$*-general* than R_2 iff for all $T_1 \in Ext(R_1)$, there exists $T_2 \in Ext(R_2)$ such that $T_2 \subseteq T_1$; R_1 is said to be *more* $\flat$*-general* than R_2 iff for all $T_2 \in Ext(R_2)$, there exists $T_1 \in Ext(R_1)$ such that $T_2 \subseteq T_1$; R_1 is said to be *strongly more* $\sharp$*-general* than R_2 iff for all rules R_3 R_1 & R_3 is more $\sharp$-general than R_2 & R_3; R_1 is said to be *strongly more* $\flat$*-general* than R_2 iff for all rules R_3 R_1 & R_3 is more $\flat$-general than R_2 & R_3. However, the cases relative to a background default theory were not considered in their approach.

Our definition of generality does not coincide with any of these notions. For instance, let p_1 and p_2 be two atoms. We have that $\top$ is both $\sharp$-general and $\flat$-general than $-p_1$ but the former is not more general than the latter in our definition. Meanwhile, $p_1 \wedge p_2$ is more general than p_1 in our definition but the former is neither strongly more $\sharp$-general nor strongly $\flat$-general than the latter. One major difference between these two approaches is that Inoue and Sakama's notions are defined based on the sets of extensions of default theories. However, two rules sharing the same set of extensions may play completely different roles in rule calculus.

Revising Default Rule Bases

Belief revision has been an important topic in solving information conflict in reasoning about agents. In most existing approaches and systems, an agent's knowledge base is usually represented by a set of classical propositional formulas, then various revision methods have been developed by researchers to solve the inconsistency by revising a knowledge base by a new piece of information.

Under the framework of rule calculus, this work can be generalized to nonmonotonic knowledge base revision. That is, in our setting, each agent's knowledge is represented as a rule base, and the problem is how to revise this rule base by giving a new default rule.

We may specify a formulation of rule base revision by generalizing approaches for propositional belief revision, for instance, the WIDTIO approach [17]. Given a rule base Δ and a rule R, we say that Δ' is a maximal subset of Δ consistent with R iff a) $\Delta' \subseteq \Delta$, b) $\Delta' \cup \{R\}$ is not a rule contradiction, and c) there does not exist Δ'' satisfying the above two conditions and $\Delta' \subset \Delta'' \subseteq \Delta$. The rule base revision operator

$\circ$ is defined as $\Delta \circ R = \bigcap \Delta' \cup \{R\}$. This revision operator also satisfies the well-known AGM postulates.

6 Conclusion

In this paper, we extend the logic of here-and-there as a general semantics for default rules. Meanwhile, we propose a corresponding axiom system for rule calculus and prove the soundness and completeness theorem (see Theorem 1). We also discuss other properties in rule calculus, including complexity issues (see Theorem 2).

The notion of strong equivalence in default logic can be directly captured in rule calculus (See Corollary 1). Corollary 1 also indicates the intuition behind rule deduction, that is, a rule R is a consequence of a rule base Δ means that R provides no more information by giving Δ. In other words, R can be eliminated by giving Δ. On the other hand, given the fact that answer set programming is a special case of default logic, our approach also shows that the logic of here-and-there and its axiomatic counterpart G3 can capture the consequence relationships among answer set programs. In fact, restricted to answer set programs (i.e., facts are atoms instead of arbitrary propositional formulas), rule calculus coincides with the notion of SE-consequence [18,19].

Rule calculus is an extension of propositional calculus and also an extension of the intermediate logic G3 [7] in the sense that the connectives in G3 are represented as rule connectives. It can also be considered as an extended logic of formalizing normality [20] since the sentence "A normally implies B" can be represented as $A \,\&\, -\neg B \Rightarrow B$ as suggested by Reiter [4].

Rule calculus is different from conditional logic [21] although both of them introduce new connectives into propositional calculus. There are two syntactic differences. First, conditional logic only introduces a conditional connective $>$. Second, whereas conditional logic allows arbitrary compositions of atoms and connectives including $>$, in most cases, it uses $>$ as a lower level connective, whilst in rule calculus, classical connectives are at the lower level. Certainly, the axiom systems and semantics of these two logics are basically dissimilar. For instance, the conditional connective $>$ is intuitively stronger than $\rightarrow$ in conditional logic, whilst the rule implication $\Rightarrow$ is, to some extent, weaker than $\rightarrow$ in rule calculus.

Another related work is so-called proof theory of default logic [22,23], which aims to define a proof-theoretical system for determining whether a propositional formula is in all (or some) extensions of a default theory. It differs from rule calculus in several aspects. Firstly, proof theory of default logic is operating on the level of extension semantics, whilst rule calculus is focused on the here-and-there semantics. Secondly, The consequence concerned in rule calculus is, in general, default rules instead of propositional formulas. Finally, even restricted to the cases of facts, as we mentioned earlier (See Proposition 2), these two systems do not coincide with each other.

Acknowledgement

This work is supported by Australian Research Council (ARC) Discovery Projects grant DP0666540. The authors thank the anonymous reviewers for their valuable comments.

References

1. Gelfond, M., Lifschitz, V., Przymusinska, H., Truszczyński, M.: Disjunctive defaults. In: Proceedings of the KR 1991, pp. 230–237 (1991)
2. Turner, H.: Strong equivalence for logic programs and default theories (made easy). In: Eiter, T., Faber, W., Truszczyński, M. (eds.) LPNMR 2001. LNCS (LNAI), vol. 2173, pp. 81–92. Springer, Heidelberg (2001)
3. Zhou, Y., Lin, F., Zhang, Y.: General default logic. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483, pp. 241–253. Springer, Heidelberg (2007)
4. Reiter, R.: A logic for default reasoning. Artificial Intelligence 13, 81–132 (1980)
5. Ben-Eliyahu-Zohary, R.: Yet some more complexity results for default logic. Artificial Intelligence 139(1), 1–20 (2002)
6. Gottlob, G.: Complexity results for nonmonotonic logics. Journal of Logic and Computation 2(3), 397–425 (1992)
7. Jongh, D.H.J.D., Hendriks, L.: Characterization of strongly equivalent logic programs in intermediate logics. Theory and Practice of Logic Programming 3(3), 259–270 (2003)
8. Cabalar, P., Ferraris, P.: Propositional theories are strongly equivalent to logic programs. Theory and Practice of Logic Programming 7(6), 745–759 (2007)
9. Ferraris, P.: Answer sets for propositional theories. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 119–131. Springer, Heidelberg (2005)
10. Pearce, D.: A new logical characterisation of stable models and answer sets. In: Dix, J., Przymusinski, T.C., Moniz Pereira, L. (eds.) NMELP 1996. LNCS, vol. 1216, pp. 57–70. Springer, Heidelberg (1997)
11. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic 2(4), 526–541 (2001)
12. van Dalen, D.: Intuitionistic logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic: Alternatives to Classical Logic, vol. III, pp. 225–339 (1986)
13. Lang, J., Liberatore, P., Marquis, P.: Propositional independence: formula-variable independence and forgetting. Journal of Artificial Intelligence Research (JAIR) 18, 391–443 (2003)
14. Plotkin, G.: A note on inductive generalization. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, pp. 153–163 (1970)
15. Muggleton, S.: Inductive logic programming. In: The MIT Encyclopedia of the Cognitive Sciences (MITECS). MIT Press, Cambridge (1999)
16. Inoue, K., Sakama, C.: Generality and equivalence relations in default logic. In: Proceedings of the AAAI 2007, pp. 434–439 (2007)
17. Winslett, M.: Updating logical databases. Cambridge University Press, Cambridge (1990)
18. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Simplifying logic programs under uniform and strong equivalence. In: Lifschitz, V., Niemelä, I. (eds.) LPNMR 2004. LNCS (LNAI), vol. 2923, pp. 87–99. Springer, Heidelberg (2004)
19. Wong, K.: Sound and complete inference rules for se-consequence. Journal of Artificial Intelligence Research 31, 205–216 (2008)
20. Geffner, H., Pearl, J.: Conditional entailment: bridging two approaches to default reasoning. Artificial Intelligence 53(2-3), 209–244 (1992)
21. Nute, D.: Conditional logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic: Extensions of Classical Logic, vol. II, pp. 387–439 (1984)
22. Bonatti, P.A., Olivetti, N.: A sequent calculus for skeptical default logic. In: Galmiche, D. (ed.) TABLEAUX 1997. LNCS, vol. 1227, pp. 107–121. Springer, Heidelberg (1997)
23. Lakemeyer, G., Levesque, H.J.: Towards an axiom system for default logic. In: Proceedings of the AAAI 2006 (2006)

Author Index