

The Duy Bui
Tuong Vinh Ho
Quang Thuy Ha (Eds.)

LNAI 5357

Intelligent Agents and Multi-Agent Systems

11th Pacific Rim International Conference
on Multi-Agents, PRIMA 2008
Hanoi, Vietnam, December 2008, Proceedings

 Springer

Lecture Notes in Artificial Intelligence 5357

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

Subseries of Lecture Notes in Computer Science

The Duy Bui Tuong Vinh Ho
Quang Thuy Ha (Eds.)

Intelligent Agents and Multi-Agent Systems

11th Pacific Rim International Conference
on Multi-Agents, PRIMA 2008
Hanoi, Vietnam, December 15-16, 2008
Proceedings

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

The Duy Bui
Quang Thuy Ha
Vietnam National University, Hanoi
College of Technology
144 Xuan Thuy, Cau Giay, Hanoi, Vietnam
E-mail: {duybt,thuyhq}@vnu.edu.vn

Tuong Vinh Ho
The Francophone Institute for Computer Science
42 Ta Quang Buu, Hai Ba Trung, Hanoi, Vietnam
E-mail: ho.tuong.vinh@auf.org

Library of Congress Control Number: 2008939922

CR Subject Classification (1998): I.2.11, I.2, C.2.4, D.2, F.3

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-89673-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-89673-9 Springer Berlin Heidelberg New York

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

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2008
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12573233 06/3180 5 4 3 2 1 0

Preface

PRIMA 2008 was the 11th in a series of conferences gathering researchers devoted to developing intelligent agents and multi-agent technologies from Asia and the Pacific regions. From its first incarnation over a decade ago, PRIMA has emerged as a significant international forum, facilitating the exchange and dissemination of innovative research from around the globe.

PRIMA 2008 was held in Vietnam, a tribute to this country's emerging scientific vitality and importance as a developing innovation center. The Program Committee received 56 submissions from 20 countries. Many of these papers are the work of PhD or Masters students from Asian countries including Korea, Japan, Indonesia, Malaysia, Iran, India, and Vietnam. In accordance with the rules, each submission was carefully peer-reviewed by three Program Committee referees. Only 19 submissions were accepted as regular papers, with a competitive rate of 33%.

Additionally, the Program Committee decided to accept 22 short papers mainly written by graduate students, allowing our young colleagues an opportunity to present their work and new perspectives. These fresh perspectives enhanced our experience of the conference and complemented the high quality of the professional papers submitted.

The array of subjects and applications explored in this year's submissions clearly demonstrate that agent technologies are now mature enough to cope with hardcore, real-world problems in a variety of domains: from pure research to applied industrial processes. As this technology matures, the potential number of real-world applications for intelligent agents and multi-agent technologies is rising dramatically. Accordingly, the number of theoretical references is also widening: foundations of multi-agent systems are now being built using game theory, mathematics, logic, software engineering, distributed systems, planning, etc. The diversity and candor of the papers selected for this edition of PRIMA are a direct reflection of the dynamic vigor of current research activity in Asia and the Pacific regions.

Special thanks are due to the Program Committee members and all the external reviewers, whose names are listed on the following pages, for their assistance in reviewing and selecting papers. We are grateful for invaluable advice from Ramakoti Sadananda, Aditya Ghose, Guido Governatori, Toru Ishida, Hiromitsu Hattori and Alexis Drogoul during the preparation of the conference. We express our appreciation to all staff of Coltech-VNUH and IFI for their active support of PRIMA 2008. Last but not least, we are grateful to Springer for its helpful collaboration and quick publication of the proceedings.

December 2008

The Duy Bui
Tuong Vinh Ho
Quang Thuy Ha

Organization

PRIMA 2008 was organized by the College of Technology, Vietnam National University, Hanoi (Coltech-VNUH) and the Francophone Institute for Computer Science (IFI).

Honorary Chairs

Huu Duc Nguyen, Vietnam
Richard Canal, Vietnam

Advisory Board

R. Sadananda, Thailand
Aditya Ghose, Australia
Guido Governatori, Australia

General Chair

Quang Thuy Ha, Vietnam

Program Co-chairs

The Duy Bui, Vietnam
Tuong Vinh Ho, Vietnam

Doctoral Mentoring Chair

Dinh Que Tran, Vietnam

Organizing Committee

The Hien Nguyen, Vietnam (Chair)
Thu Hien Tran, Vietnam
Chau Mai, Vietnam
Oanh Tran, Vietnam
Huong Thao Nguyen, Vietnam

Publicity Chair

Son Bao Pham, Vietnam

Program Committee

Abdul Sattar, Australia
Aditya Ghose, Australia
Alan Liu, Taiwan
Alberto Fernandez, Spain
Alexis Drogoul, Vietnam
Antonino Rotolo, Italy
Chao-Lin Liu, Taiwan
Dencho Batanov, Cyprus
Dinh Que Tran, Vietnam
Dirk Heylen, The Netherlands
Dongmo Zhang, Australia
Frank Dignum, The Netherlands
Graham Low, Australia
Guido Governatori, Australia
Hans van Ditmarsch, New Zealand
Ho-fung Leung, China
Jaeho Lee, Korea
Jean-Luc Koning, France
Jane Hsu, Taiwan
Joerg Denzinger, Canada
Joongmin Choi, Korea
Jyi-Shane Liu, Taiwan
Jung-Jin Yang, Korea
Kamal Karlapalem, India
Leendert van der Torre, Luxembourg
Lin Liu, China
Mehmet Orgun, Australia
Michael Winikoff, Australia
Mike Barley, New Zealand
Minkoo Kim, Korea
Naoki Fukuta, Japan
Ngoc Thanh Nguyen, Poland
Nicolas Marilleau, Vietnam
Paolo Giorgini, Italy
Rafael H. Bordini, UK
Ryszard Kowalczyk, Australia
Shaheen Fatima, UK
Shivashankar B. Nair, India
Son Bao Pham, Vietnam
Stephane Bressan, Singapore
Suresh Chande, Finland
Takayuki Ito, Japan
Tokuro Matsuo, Japan
Tommie Meyer, South Africa

Toru Ishida, Japan
Toshiharu Sugawara, Japan
Valentin Robu, The Netherlands
Vineet Padmanabhan, India
Virginia Dignum, The Netherlands
Von-Wun Soo, Taiwan
Wayne Wobcke, Australia
Wojtek Jamroga, Germany
Yasuhiko Kitamura, Japan
Zili Zhang, Australia

External Reviewers

Huiye Ma
Ji Ma

Jie Tang
Motohiro Mase

Table of Contents

Keynote Speech: A Review of the Ontological Status, Computational Foundations and Methodological Processes of Agent-Based Modeling and Simulation Approaches: Open Challenges and Research Perspectives	1
<i>Alexis Drogoul</i>	
Keynote Speech: Computational Collective Intelligence and Knowledge Inconsistency in Multi-Agent Environments	2
<i>Ngoc Thanh Nguyen</i>	
Keynote Speech: Agent Oriented Software Engineering: Why and How	4
<i>Lin Padgham</i>	
Coordinating Agents Plans in Multi-Agent Systems Using Colored Petri Nets	6
<i>Maryam Noorae Abadeh, Kamran Zaminifar, and Mohammad-Reza Khayyambashi</i>	
Design of an Internet-Based Advisory System: A Multi-Agent Approach	14
<i>Saadat M. Alhashmi</i>	
Towards Virtual Epidemiology: An Agent-Based Approach to the Modeling of H5N1 Propagation and Persistence in North-Vietnam	26
<i>Edouard Amouroux, Stéphanie Desvoux, and Alexis Drogoul</i>	
Measurement of Underlying Cooperation in Multiagent Reinforcement Learning	34
<i>Sachiyo Arai, Yoshihisa Ishigaki, and Hironori Hirata</i>	
Reo Connectors as Coordination Artifacts in 2APL Systems	42
<i>Farhad Arbab, Lăcrămioara Aștefănoaei, Frank S. de Boer, Mehdi Dastani, John-Jules Meyer, and Nick Tinnermeier</i>	
A Verification Framework for Normative Multi-Agent Systems	54
<i>Lăcrămioara Aștefănoaei, Mehdi Dastani, John-Jules Meyer, and Frank S. de Boer</i>	
Social Viewpoints for Arguing about Coalitions	66
<i>Guido Boella, Leendert van der Torre, and Serena Villata</i>	
Changing Institutional Goals and Beliefs of Autonomous Agents	78
<i>Guido Boella, Leendert van der Torre, and Serena Villata</i>	

Reasoning about Constitutive Norms, Counts-As Conditionals, Institutions, Deadlines and Violations	86
<i>Guido Boella, Jan Broersen, and Leendert van der Torre</i>	
When to Use a Multi-Agent System?	98
<i>Paul Bogg, Ghassan Beydoun, and Graham Low</i>	
Multiagent Incremental Learning in Networks	109
<i>Gauvain Bourgne, Amal El Fallah Seghrouchni, Nicolas Maudet, and Henry Soldano</i>	
UML-F in the Design of an Agent-Oriented Software Framework	121
<i>Daniel Cabrera-Paniagua and Claudio Cubillos</i>	
Interactive Learning of Expert Criteria for Rescue Simulations	127
<i>Thanh-Quang Chu, Alain Boucher, Alexis Drogoul, Duc-An Vo, Hong-Phuong Nguyen, and Jean-Daniel Zucker</i>	
Modularity in Agent Programming Languages: An Illustration in Extended 2APL	139
<i>Mehdi Dastani, Christian P. Mol, and Bas R. Steunebrink</i>	
On the Pheromone Update Rules of Ant Colony Optimization Approaches for the Job Shop Scheduling Problem	153
<i>Dong Do Duc, Huy Q. Dinh, and Huan Hoang Xuan</i>	
Preliminary Result on Secure Protocols for Multiple Issue Negotiation Problems	161
<i>Katsuhide Fujita, Takayuki Ito, and Mark Klein</i>	
Performance Analysis about Parallel Greedy Approximation on Combinatorial Auctions	173
<i>Naoki Fukuta and Takayuki Ito</i>	
Online Market Coordination	185
<i>Masabumi Furuhata, Dongmo Zhang, and Laurent Perrussel</i>	
Towards an Evaluation Framework for MAS Software Engineering	197
<i>Emilia Garcia, Adriana Giret, and Vicente Botti</i>	
From Obligations to Organizational Structures in Multi-Agent Systems	206
<i>J. Octavio Gutiérrez-García, Jean-Luc Koning, and Félix F. Ramos-Corchado</i>	
Addressing the Brittleness of Agent Interaction	214
<i>Mohd Fadzil Hassan and Dave Robertson</i>	
Dividing Agents on the Grid for Large Scale Simulation	222
<i>Dac Phuong Ho, The Duy Bui, and Nguyen Luong Do</i>	

An Interest Rate Adjusting Method with Bayesian Estimation in Social Lending	231
<i>Masashi Iwakami and Takayuki Ito</i>	
A Temporal Logic for Stochastic Multi-Agent Systems	239
<i>Wojciech Jamroga</i>	
Applying the Logic of Multiple-Valued Argumentation to Social Web: SNS and Wikipedia	251
<i>Shusuke Kuribara, Safia Abbas, and Hajime Sawamura</i>	
Simulation of Halal Food Supply Chain with Certification System: A Multi-Agent System Approach	259
<i>YiHua Lam and Saadat M. Alhashmi</i>	
A Novel Approach for Conflict Resolution in Context-Awareness Using Semantic Unification of Multi-Cognition	267
<i>Keonsoo Lee and Minkoo Kim</i>	
Improving Trade-Offs in Bilateral Negotiations under Complete and Incomplete Information Settings	275
<i>Ivan Marsa-Maestre, Miguel A. Lopez-Carmona, and Juan R. Velasco</i>	
PAMS – A New Collaborative Framework for Agent-Based Simulation of Complex Systems	287
<i>Trong Khanh Nguyen, Nicolas Marilleau, and Tuong Vinh Ho</i>	
Methodological Steps and Issues When Deriving Individual Based-Models from Equation-Based Models: A Case Study in Population Dynamics	295
<i>Ngoc Doanh Nguyen, Alexis Drogoul, and Pierre Auger</i>	
Abstraction of Agent Cooperation in Agent Oriented Programming Language	307
<i>Nguyen Tuan Duc and Ikuo Takeuchi</i>	
Knowledge Assessment: A Modal Logic Approach	315
<i>Vineet Padmanabhan, Guido Governatori, and Subhasis Thakur</i>	
The Design of a Self-locating Automatic-Driving Robot	323
<i>Gi-Duck Park, Robert McCartney, and Jung-Jin Yang</i>	
Settling on the Group’s Goals: An n-Person Argumentation Game Approach	328
<i>Duy Hoang Pham, Subhasis Thakur, and Guido Governatori</i>	
Revenue Maximising Adaptive Auctioneer Agent	340
<i>Janine Claire Pike and Elizabeth Marie Ehlers</i>	

Managing Collaboration Using Agent Based Simulation	348
<i>Utomo Sarjono Putro, Manahan Siallagan, Santi Novani, and Dhanan Sarwo Utomo</i>	
Using Agent-Based Simulation of Human Behavior to Reduce Evacuation Time	357
<i>Arief Rahman, Ahmad Kamil Mahmood, and Etienne Schneider</i>	
Participatory Simulation Platform Using Network Games	370
<i>Shoichi Sawada, Hiromitsu Hattori, Marika Odagaki, Kengo Nakajima, and Toru Ishida</i>	
A Multiagent-System Framework for Hierarchical Control and Monitoring of Complex Process Control Systems	381
<i>Vu Van Tan, Dae-Seung Yoo, and Myeong-Jae Yi</i>	
Agent Reasoning with Semantic Web in Web Blogs	389
<i>Dinh Que Tran and Tuan Nha Hoang</i>	
Design of a Multiagent System over Mobile Devices for the Planning of Touristic Travels	397
<i>Miguel Valdés and Claudio Cubillos</i>	
Author Index	405

Keynote Speech: A Review of the Ontological Status, Computational Foundations and Methodological Processes of Agent-Based Modeling and Simulation Approaches: Open Challenges and Research Perspectives

Alexis Drogoul

IRD, UR079-GEODES, 32 av. Henri Varagnat, 93143 Bondy Cedex, France
alexis.drogoul@gmail.com

Agent based modeling (ABM) and simulation techniques are now used in a number of domains, ranging from social sciences, ecology or biology to exact sciences like physics. They provide modelers with the ability to understand and reproduce, through virtual experiments, the emergence of nearly any kind of macro-structure or macro-dynamics from the interactions of lower level computer programs called agents. As these agents can be programmed with all the details required and can arbitrarily cover any level of representation, ABM undeniably represents one of the most versatile approaches to understanding complex systems through simulation.

However, a lot of criticisms have also been expressed against them, which roughly fall into three categories:

- Criticisms regarding the ontological status of agent-based models in scientific research and that of the results they provide.
- Criticisms (not unrelated to the previous ones) regarding the difficulty of design, calibration, manipulation and validation of these models.
- Criticisms regarding their apparent inability to cope with multiple levels of abstraction at once, in both modeling and simulation.

My talk will try to reply to these criticisms in the light of successful achievements of ABMs, but also show that most of them, although they can concern other modeling and simulation approaches, are effectively sound. This will give me the opportunity to underline which current and future directions of multi-agent researches should be mobilized in order to provide innovative answers and therefore contribute to the reinforcement and credibility of ABMs, especially with respect to the last category of criticisms.

Keynote Speech: Computational Collective Intelligence and Knowledge Inconsistency in Multi-agent Environments

Ngoc Thanh Nguyen

Institute of Computer Science, Wrocław University of Technology
Str. Janiszewskiego 11/17, 50-370 Wrocław, Poland
thanh@pwr.wroc.pl

It is well-known that the knowledge of a group is not the same as the union of knowledge states of the group members. Let's consider several two examples. If an agent A knows that $a > b$ and if an agent B knows that $b > c$ for some real numbers variables a , b and c , then together these agents know not only $a > b$ and $b > c$, but also $a > c$. If, in another case, A knows that $a \geq b$ and B knows that $b \geq a$ then together they know also that $a = b$. One can then state that the power of the collective knowledge exists owing to the inconsistency of knowledge of its members.

Computational Collective Intelligence (CCI) is understood as an AI sub-field dealing with soft computing methods which enable making group decisions or processing knowledge among autonomous units acting in distributed environments. Web-based systems, social networks and multi-agent systems very often need these tools for working out consistent knowledge states, resolving conflicts and making decisions.

In this talk I present several aspects related to the answers of the following questions:

1. How to determine the knowledge of a collective on the basis of the knowledge of its members?
2. How to evaluate the quality of the collective knowledge?
3. Is the intelligence of a collective larger than the intelligence of its members?
4. How multi-agent systems can use these results?

Many examples show that the knowledge of a collective is not a usual "sum" of the knowledge of its members. For multi-agent environments, in case of some conflict between agents referring to the proper knowledge of a real world, CCI tools may be very useful for reconciling the inconsistency and processing the knowledge of agents.

In the first part of this talk, the theoretical foundation for conflict analysis and consensus choice is presented. The inconsistency of knowledge at syntactic and semantic levels and the proposal of the structures for representing inconsistency and algorithms for its solution are analyzed. The analysis of expert knowledge inconsistency and conflict of ontologies is also included. It has been shown that that inconsistency of experts is very useful. In work [1] the author defined and analyzed classes of consensus functions. He also studied such conceptions as consensus susceptibility, quality of consensus, or reduction of consensus numbers. Besides, he introduced a model for knowledge integration and analyzed it to great theoretical

details. In addition, he has worked out an algorithm for integration, which is quite useful to put the theory in practice by implementing it as a working system. The author has shown methods for knowledge inconsistency resolution and knowledge integration both on the syntactic and semantic levels. Furthermore, he has investigated more practical issues such as a method of inconsistency resolution for ontologies, a method for reconciling inconsistency of knowledge of experts, a method for determining a learning scenario in intelligent tutoring systems, and a method for reconciling inconsistent knowledge and answers given by different agents in a meta-search engine.

In the second part, a formal model for calculating the knowledge of collectives with using a quasi-Euclidean space is presented. The original features of this model are based on the proofs of several results owing to which one can get to know about the relationships between the collective knowledge state and the real knowledge state. These results show that the knowledge of a collective is more proper than the knowledge of its members. Thus in some restricted scope one can state that the hypothesis “*A collective is more intelligent than one its single member*” is true. Some relationships between the consistency degree of a collective and the distance between the collective knowledge state and the real knowledge state are also presented. As it has been proved, the assumption of identical distances between collective members and the real knowledge state may improve the quality of the collective knowledge state in the sense that it is more similar to the proper knowledge state than each of the members’ states [2].

In the third part, several approaches of using collective knowledge determination are proposed to be applied in multi-agent environments. Two applications based on the theoretical work are presented. The first refers to recommendation in intelligent tutoring system and the second is about creating a consensus-based multi-agent system to aid users in information retrieval from the Internet [3].

References

1. Nguyen, N.T.: *Advanced Methods for Inconsistent Knowledge Management*. Springer, London (2008)
2. Nguyen, N.T.: Inconsistency of Knowledge and Collective Intelligence. *Cybernetics and Systems* 39(6), 542–562 (2008)
3. Sliwko, L., Nguyen, N.T.: Using Multi-agent Systems and Consensus Methods for Information Retrieval in Internet. *International Journal of Intelligent Information and Database Systems* 1(2), 181–198 (2007)

Keynote Speech: Agent Oriented Software Engineering: Why and How

Lin Padgham

RMIT
University Melbourne,
Australia

Intelligent agents are a new paradigm for developing a wide range of software systems. Just as object oriented programming is at a higher level of abstraction than procedural programming, so agent oriented is at a higher abstraction level than object oriented. This facilitates faster and easier development of more complex systems, than is possible with a less powerful paradigm. Some research has shown that efficiency gains of more than 300 oriented approach to development.

One of the most popular agent approaches is what is known as the Belief Desire Intention (BDI) agent paradigm. It is this approach which was the basis of the efficiency gains mentioned above. This approach, at the practical, engineering level, is based on the notion of goals, and abstract plans as ways to achieve these goals. Goals can typically be achieved in some number of different ways, even at the most abstract level. These different ways may be applicable only in certain situations. A plan is thus an abstract description for achieving a goal, in a specified situation (or context). The executable plan body is made up of a combination of actions and sub-goal steps. (Depending on the system, plans can also contain arbitrary code, in addition to subgoals and actions). The subgoals then also have plans which realise them. If we start with a single goal, and assume that:

- each goal has at least two plans that describe how to achieve it
- each plan contains at least three subgoals
- there is an abstraction depth of four

then there are over a million ways to achieve that single goal. This is incredibly powerful! Also, the way in which these systems expand plans as needed, combined with an ability to try alternative means to achieve a (sub) goal if a plan fails, means the system is very robust to environmental change during execution. This makes them extremely well suited to systems situated in dynamic environments, although this characteristic is by no means necessary, in order to realise the benefits of the flexibility and modularity inherent in the paradigm.

The inherent modularity of these kind of systems also makes them very suitable for evolving over time - an increasingly common phenomenon for complex ICT systems. It is very common that a core system is developed quickly, but then additions and extensions are added once the system is in use. Some small experimental work we did some years ago, showed that modification of a

program was both easier and more localised, when an agent paradigm was used, than when the program was developed using a traditional approach.

However in order to realise these advantages of the agent paradigm, systems must be designed and developed in appropriate ways. This requires engineers and software developers to think differently about the design, as compared to Object Oriented approaches. There are of course many common principles. But there are also aspects that require a different way of thinking.

Recognition of this need to think differently when using an agent paradigm to build systems (combined with the frustration of seeing many student programs that gained no advantage from the agent approach), led to the development of the Prometheus methodology for developing Agent Systems. This was developed by myself and colleagues at RMIT, in partnership with Agent Oriented Software, who also needed such a methodology to assist their clients and customers in use of their agent platform product, JACK. The Prometheus methodology has developed over more than ten years, and was one of the earliest detailed approaches to building agent systems. Today there are a large number of Agent Oriented Software Engineering methodologies, and an international journal devoted entirely to this topic.

In this talk we will present a basic overview of the Prometheus methodology, and some of the design models produced using this approach to designing an agent system. Recent work comparing some of the most prominent approaches - Tropos, O-MASE and Prometheus - on a common case study, has highlighted the substantial common core within the approaches developed by these different groups. There is current activity in standards bodies to try and standardise various aspects of agent modelling, and it is possible that the next few years will see further movement in this direction.

As the core of Agent Oriented Software Engineering coalesces somewhat, there is also further development of various aspects such as design of teams, social structures and organisations; automated testing and debugging; aspect oriented development; and support for maintenance, amongst others. We will review briefly some of these developments and directions.

Coordinating Agents Plans in Multi-Agent Systems Using Colored Petri Nets

Maryam Noorae Abadeh¹, Kamran Zaminifar²,
and Mohammad-Reza Khayyambashi³

¹ Department of Computer Engineering, Islamic Azad University,
Abadan and Khorramshahr Branch, Khuzestan, Iran

² Department of Computer Engineering, Islamic Azad University of Najaf Abad, Isfahan, Iran

³ Computer Department, Faculty of Engineering, University of Isfahan, Isfahan, Iran
shadab_noorae@yahoo.com, zamanifar@eng.ui.ac.ir,
m.r.khayyambashi@eng.ui.ac.ir

Abstract. Applying coordination mechanisms to handle interdependencies that exist between agents in multi-agent systems (MASs), is an important issue. In this paper, two levels MAS modeling scheme and a language to describe a MAS plan based on interdependencies between agents' plans are proposed. Initially a generic study of possible interdependencies between agents in MASs is presented, followed by the formal modeling (using Colored Petri Nets) of coordination mechanisms for those dependencies. These mechanisms control the dependencies between agents to avoid unsafe interactions where individual agents' plans are merged into a global multi-agent plan. This separation, managed by the coordination mechanisms, offers more powerful modularity in MASs modeling.

Keywords: Agent Plan, Colored Petri Nets, Coordination Mechanism, Dependency, Multi-Agent System.

1 Introduction

There are two main streams of research on software agents, namely the multi-agent systems (MAS) and mobile agents (MA). Research on multi-agent systems (MAS) is rooted in distributed artificial intelligence, and dates back to the fifties [3]. In a multi-agent system, agents are autonomous, reactive, proactive and sociable. The idea of agents as autonomous entities, which can interact with each other to solve problems, has led to the development of interests in agent-based design paradigm for software engineering [1].

A multi-agent system can be considered as a loosely coupled network of problem solver entities that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity [15].

Try to ensure that all members of the system act consistently, is critical in the MAS design [4].

In its general terms, coordination is the art of managing interactions and dependencies among activities, that is, among agents, in the context of MASs [5, 6].

The coordination is highly dynamic, because it needs to be negotiate again almost continuously during a collaborative effort. In other words, coordination policies change according to the cooperation instance, or even during the evolution of the same instance. Therefore, it is essential that coordination mechanisms be flexible enough to handle the flexibility demanded by the dynamics of the communications among the agents [7].

Agents can improve consistency by planning their actions. Planning for a single agent is a process of constructing a sequence of actions considering only capabilities, environmental constraints and goals. When writing a multi-agent plan, the syntax is not much different from the single agent case, except that actions are labeled with a unique id for the agent [24] and coordination constraints are applied. In our approach, is focused on modeling MAS plan, from local agent plans with concerning to interdependencies between individual plans.

Describing multi-agent plans has been considered as *plan sharing* or *centralized planning*, where the objective is to distribute a global plan to agents executing them, or as *plan merging*, where individual plans are merged into a global plan with considering coordination issues [15]. In particular, we want to be able to order actions across plans so that overall consistency is maintained and conflicting situations are avoided [24]. Figure 1 clearly, describe MAS planning architecture. In this paper is focused on plans merging section and applying coordination policies.

The main idea is to separate the coordination mechanisms from the MAS modeling where the multi-agent centralized plan from individual agents plans is made. In our proposed schema, to design a MAS plan, at the top level are provided a set of local agents plans without a great deal of detail coordination mechanisms. At the lower level is provided a high degree of modeling detail in the form of Colored Petri Nets that represent the global multi-agent system plan. In this way multi-agent systems modeling, offer more powerful modularity. Also it is defined a modeling language for specifying, modeling and documenting systems that describe the MAS global plan. Using the language it is possible to automate MAS model building, just using the description dependencies based file.

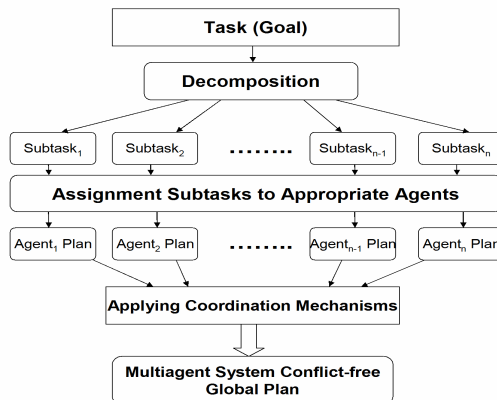


Fig. 1. MAS Planning Approach

To model the proposed coordination mechanisms, we use an approach based on Colored Petri Nets. Petri Nets combine a well defined mathematical theory with a graphical representation the dynamic behavior of systems. Furthermore, CPNs offer a strong theoretical support for the analysis of an environment's behavior and supplementary simulation techniques. We use this tool to anticipate and test the behavior of multi-agent systems before their implementation. For more information, some references are [8, 9, 19, and 20].

This paper is structured as follows; section 2 discusses related research work. Section 3 presents our agents plans coordination approach. Coordination mechanisms modeled using CPN are presented in section 4. Section 5 discusses a language to describe MASs plans. Section 6 presents a conclusion and mentions future works.

2 Related Work

Applying Petri Nets to MAS modeling has been an active research. [13] and [14] are two papers that deal with agent behavior coordination. For example in [13], is introduced a moderator coordination model to handle agents communication from the organization level. A set of Petri net based moderators, each managing a single conversation, and a conversation server, which organizes moderators, are the moderator parts. In contrast, our work does not draw a clear boundary along conversations, but instead focuses on agent behavior from the view of an individual agent and from the global view of the whole MAS; this provides a powerful modular modeling. Concerning Petri net based MAS modeling, there also exists research work emphasizing special modeling aspects [21], [22], [23] and [24], but with little emphasis on agent behavior coordination. For example, [22] emphasizes model verification to check whether a multi-agent plan is possible. Many existing research works focus on a high level interpretation of agents and MASs as Petri Net models in a whole [10] and [11]. Such design does not address further details about agent coordination. Our work is more generic than those presented above. We define a set of interdependencies among tasks in different agents' plans and associated coordination mechanisms (modeled using high level PNs) that can be used in multi-agent systems, multi-user interaction and virtual environments. Also a language to describe a MAS global plan is described that is a description of a MAS model based on interdependencies between agents.

3 Coordinating Agents Plans

As already stated, this work intends to provide a two layers approach for modeling MASs using CPN-based mechanisms to manage interdependencies among collaborative problem solvers in MASs and guarantee that these dependencies will not be violated. The idea is that the designer of a multi-agent system will be concerned only with the definition of tasks that any agent does in its local plans and their interdependencies with the tasks in other agents' plans, and not with the management of those dependencies.

In the proposed schema, a MAS is modeled in two distinct levels. In the first level, the tasks in the agents' individual plans are defined and their interdependencies types

are expressed. In the next level, interdependent tasks are expanded and the adequate coordination mechanisms are inserted among them. During the passage from the local plans to the global plan designing, each task in any agent's workflow which has interdependency with another is expanded according to the model of Figure 2 [16].

According to [26], the five places associated to the expanded tasks represent the interface with the resource manager and a task that executes in any agent plan is explained. Places *request_resource*, *assigned_resource* and *release_resource* associate any task with resource manager and places *execute_task* and *finish_task* indicate, respectively, the beginning and the end of the task.

Using CPN, it is possible to share simulation places and using colored token, different tasks are distinguished.

The main goal of our work is to construct the multi-agent global plan from individual local plans with no more focus on interdependencies, because once the interdependencies are defined, the expansion of tasks in any agent workflow is according to the model of Figure 2 and the insertion of coordination mechanisms can be automated. We use two general classes of interdependencies that have been expressed in [17]: temporal and resource management.

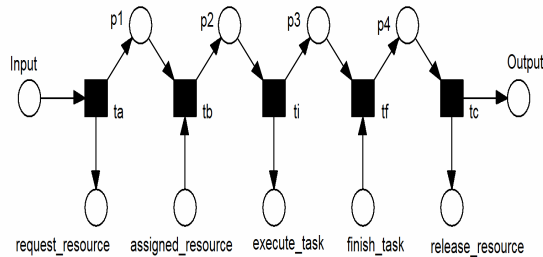


Fig. 2. An expanded task in the local plan level

Temporal interdependencies establish relative order of execution between a pair of tasks that exist in two different agents local plans and these types of dependencies establish execution order of tasks. Coordination mechanisms associated to this kind of dependency have *execute_task* as input place and *finish_task* as output place (Figure 2).

Exclusive relations between time intervals [17], has been used to define temporal dependencies between tasks in different agents local plans in MASs [26]. For example relation *finish* between *taskA* in *agent1 plan* and *taskB* in *agent2 plan* define that both tasks must end together and if it is necessary that *taskA* start before *taskB*, the relation is named *finishA*, and if it is not matter which task starts before, it is called *finishB* or relation *taskA during taskB* is defined that two variations are possible. In the first one (*duringA*), *taskB* can be executed only once during the execution of *taskA*. In the second one (*duringB*), *taskB* can be executed several times.

In [17], three basic mechanisms for resource management are defined: *Sharing*, *Simultaneity*, *Volatility*. For example *Simultaneity* indicates resource is available only if a certain number of agents request it simultaneously.

In the following, the models for the coordination mechanisms related to *finishA* and *sharing* dependencies are shown. The other dependencies are similarly modeled.

4 Modeling Coordination Mechanisms Using CPNs

However, a typical problem in the use of ordinary PN is the state explosion, which can occur in our context when the number of interdependencies increases. CPNs reduce this problem because they generate simpler models, with less places and transitions. Therefore, we modeled those using CPNs. In temporal dependencies models we focus on a relevant piece of extended task model include *execute_task* and *finish_task* places.

The transitions called *LogicA* and *LogicB* represent the logistic of the tasks. They are represented as non-instantaneous transitions with token reservation.

Figure 3 shows the coordination mechanism for the relation *TaskB finishA TaskA* (with restriction on which task should start at the first). In the Figure, the arcs $1^{\setminus}JobA++1^{\setminus}JobB$ ensure that two different tasks (tokens of different colors) are available at the same time. Transition *t2* ensure ending tasks concurrently. In order to model the relation with the restriction that *TaskB* must start after *TaskA* (*finishA*) it is necessary to dependence *LogicB* to the end of *LogicA*, ensuring that a token will be sent to transition *LogicB* after the end of *TaskA*.

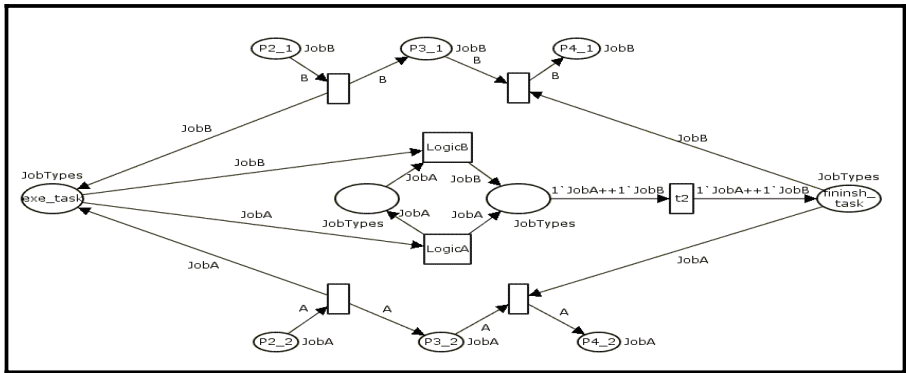


Fig. 3. Coordination model for Relation *TaskB finishA TaskA*

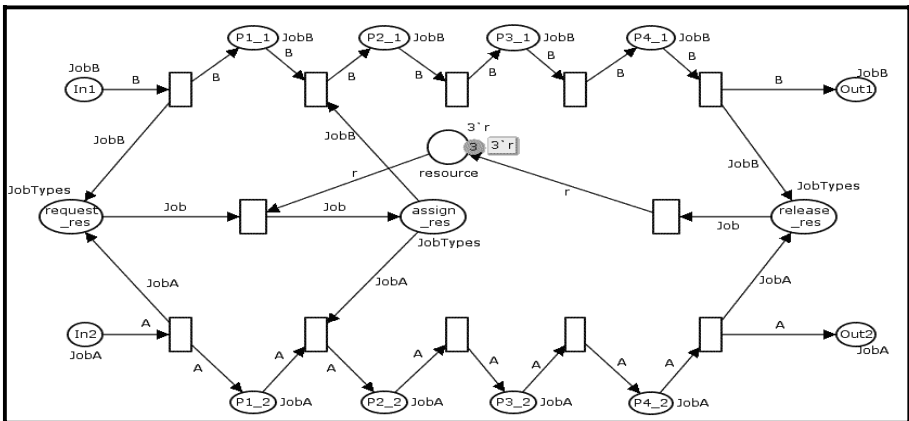


Fig. 4. Coordination model for Relation *sharing*

Figure 4 shows the mechanism for *sharing*. 3 resources (tokens r) are going to receive if they are available in place *res*.

4.1 Simulation

Colored Petri Nets and Design/CPN Tools have been used for modeling, execution and validation analysis to test these models, automatically or interactive. Design/CPN Tools is an industrial-strength computer tool for building and analyzing CPN models. Using CPN Tools, the behavior of these models using simulation, to verify properties via state space methods was investigated. Also in this paper, validation analysis method is used. Iterative simulations of fictitious cases were also examined, to ensure that the model treats them correctly.

4.2 Generalization

It is possible to generalize the modeling approach. It can have different interpretations:

- These mechanisms can have expanded from "two" to "many" (more than two) agents plans (Jobs). It means that N task in M local plans can have temporal or (and) resource dependencies. For example taskA in agent1 plan and taskB in agent2 plan and taskM in agentN plan have *starts* relation.
- Two tasks in two different agents' plans can have more than one dependency. Thus it is possible to composite mechanisms from the basic ones discussed above. For example, taskA in agent1 plan and taskB in agent2 plan have *startA + duringB + simultaneity* relations or *sharing M + volatility N* indicates that up to M tasks may share the resource, which can be used N times only.

5 A Language for Definition Interdependencies of MAS Plan

A language that is corresponding to the MAS plan Model, based on interdependencies between agents and coordination mechanism is presented. Building MAS model from this descriptive language is a benefit of this language; also it is possible to automate the passage from the first level to the coordination level of the MAS modeling using these descriptive statements. A dependency is defined as according to the following syntax:

AgentName_i.TaskName [Dependency Kind] AgentName_j.TaskName

It is necessary that tasks have the same name as the transitions representing them in the local agents' plans. When all agent dependencies are described, these set of statements define the MAS model (global plan).

6 Conclusion

The coordination of interdependent activities in multi-agent systems is a problem that should be addressed to ensure the effectiveness of the cooperation. The separation

between individual agents' plans and dependencies with other agents' plans, and the utilization of reusable coordination mechanisms are steps towards this goal.

Petri Nets, due to their support for modeling, simulation and analysis, have proven to be a powerful tool for verifying the correctness and validating the effectiveness of collaborative environments before their actual implementation [2, 12].

One of the next steps of this work is to automate the passage from the individual plan to the sharing plan of models in a CPN simulation tool.

References

1. Jennings, N.R.: An Agent-Based Approach for Building Complex Software Systems. *Communications of ACM* 44(4), 35–41 (2001)
2. Kinny, D., Georgeff, M.P.: Modeling and Design of Multi-Agent Systems. In: Rao, A., Singh, M.P., Wooldridge, M.J. (eds.) *ATAL 1997*. LNCS, vol. 1365, pp. 1–20. Springer, Heidelberg (1998)
3. Green, S., Hurst, L., Nangle, B., Cunningham, P., Somers, F., Evans, R.: Software Agents: A Review. In: Intelligent Agent Group (IAG) report TCD-CS-1997-06, Trinity College Dublin (1997)
4. Jennings, N.R.: Coordination Techniques for Distributed Artificial Intelligence. In: O'Hare, G.M.P., Jennings, N.R. (eds.) *Foundation of Distributed Artificial Intelligence, Sixth-Generation Computer Technology Series*, pp. 187–210. Wiley, New York (1996)
5. Ciancarini, P.: Coordination models and languages as software integrators. *ACM Computing Surveys* 28(2) (1996)
6. Gelernter, D., Carriero, N.: Coordination languages and their significance. *Communications of the ACM* 35(2), 97–107 (1992)
7. Edwards, W.K.: Policies and Roles in Collaborative Applications. In: *ACM Conf. on Computer Supported Cooperative Work*, pp. 11–20 (1996)
8. Murata, T.: Petri nets: properties, analysis and applications. *Proceedings of the IEEE* 77(4), 541–580 (1999)
9. CPN Tools, <http://www.daimi.au.dk/CPNTools>
10. Holvoet, T.: Agents and Petri Nets. *Petri Net Newsletters* 49 (1995)
11. Moldt, D., Wienberg, F.: Multi-Agent-Systems Based on Coloured Petri Nets. In: *Proceedings of 18th International Conference on Application and Theory of Petri Nets* (1997)
12. Shoham, Y.: Agent-Oriented Programming. *Artificial Intelligence* 60, 51–92 (1993)
13. Hanachi, C., Blanc, C.S.: Protocol Moderators as Active Middle-Agents in Multi-Agent Systems. *Autonomous Agents and Multi-Agent Systems* 8(2), 131–164 (2004)
14. Weyns, D., Holvoet, T.: A Colored Petri Net for Regional Synchronization in Situated Multi-Agent Systems. In: *Proceeding of First International Workshop on Petri Nets and Coordination (PNC)*, Bologna, Italy, June 21–26 (2004)
15. Durfee, E.: Distributed problem solving and planning. In: Weiss, G. (ed.) *Multi-agent Systems: a Modern Approach to Distributed Artificial Intelligence*, pp. 121–164. MIT Press, Cambridge (1999)
16. Van der Aalst, W.M.P.: Modelling and analyzing workflow using a Petri-net based approach. In: *Proceedings 2nd Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pp. 31–50 (1994)
17. Raposo, A.B., Magalhães, L.P., Ricarte, I.L.M.: Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments. *Int. J. of Computer Systems Science & Engineering* (2000)

18. Allen, J.F.: Towards a General Theory of Action and Time. *Artificial Intelligence* 23, 123–154 (1984)
19. Jensen, K.: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Basic Concepts*, vol. 1. Springer, Heidelberg (1992)
20. Examples of Industrial Use of CP-nets, <http://www.daimi.au.dk/CPnets/intro/exampleindu.html>
21. Jennings, N., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems Journal* 1(1), 7–38 (1998)
22. Xu, D., Volz, R.A., Ioerger, T.R., Yen, J.: Modeling and Analyzing Multi-agent Behaviors Using Predicate/transition Nets. *International Journal of Software Engineering* 13(1), 103–124 (2003)
23. Murata, T., Nelson, P.C., Yim, J.: Predicate-Transition Net Model for Multiple Agent Planning. *Information Science* 57/58, 361–384 (1991)
24. Vittorio, A., Ziparo and Luca, I.: Petri Net Plans. In: *Fourth International Workshop on Modelling of Objects, Components, and Agents* (2006)
25. Sycara, K.: Multi-agent Systems. *Intelligent Agents AI magazine* 19(2) (Summer 1998)
26. Noorae, M., Zamanifar, K.: Modeling Multiagent Systems Using Colored Petri Nets. In: *IADIS International Conference Intelligent Systems and Agents*, pp. 85–91 (2008)

Design of an Internet-Based Advisory System: A Multi-agent Approach

Saadat M. Alhashmi

School of Information Technology, Sunway Campus, Monash University,
46150, Selangor, Malaysia
saadat.m.alhashmi@infotech.monash.edu.my

Abstract. With the emerging proliferation of information and communications technology in the home and work environments, the provision of computer-based medical advisory systems in healthcare could lead to huge savings to the cost of caring for patients with chronic conditions, such as, diabetes, asthma and hypertension. This paper proposes that an internet-based medical expert system could facilitate a far more efficient system in eliminating the number of unnecessary visits to General Practitioner (GP) for routine consultations. An internet-based intelligent system implementing a variety of functions carried in GP consultations is, thus proposed. The system design is based on multi-agent architecture, which attempts to replicate the roles of each person in a typical GP consultation environment. The role of clinical decision-making is carried out by a fuzzy inference engine linked to a knowledge-base of patient records. The management of diabetes is presented as a case study in the paper.

Keywords: Multi-agent, advisory system, diabetes, fuzzy logic.

1 Introduction

The information explosion has brought abundant benefits with it, but in the hindsight it also brought an urge and an urgency to make decisions in real time. This decision-making could be in the form of information retrieval, finding the best deal for an excursion, ordering something or asking an expert for advice. Fuzzy Logic, Neural Networks, for example, are but a few of the tools within artificial intelligence that have found many applications to facilitate decision-making.

In this paradigm, a user is required to specify their query, which is then compared with the rule-base or 'built in intelligence'. In spite of great advancements in the efficiency of decision-making tools, most of these tools can further be improved. Based on the literature review, it was envisaged that a system depicting characterisation of human intelligence such as autonomy and co-operation would immensely help in demonstrating the task delegation to computers.

Another interesting domain is the multi-agent environment that stemmed mainly from the distributed computing and artificial intelligence community. This paper picks up the same theme from a different perspective, that is, it looks into proposing and constructing a system, which can be a generic model for seeking expert advice in a distributed environment using multi-agents. The proposed work will be exemplified

on one of the many issues in healthcare domain. The rationale is also to assess the improvement in performance in this domain in particular and other domains in general.

Healthcare domain choice was influenced by several factors. There has been growing pressure on the healthcare industry to cut down costs and improve quality. An ageing population, declining death rates and improved standards of living have added to this pressure. Furthermore, the delicate and highly sensitive nature of this domain requires that there should be absolutely no compromise on quality while looking for a reduction in costs. We also hold the view that a system should incorporate the features of autonomy and co-operation, and expert knowledge. Some of the many functions it could help in facilitating are: Autonomous negotiation, Task delegation, Keeping-an-eye functionality, exploitation of expert knowledge for decision-making.

In this paper, a co-operative autonomous decision-making system is proposed with autonomy and co-operation facets of human intelligence. Fuzzy logic is used to depict decision-making and the system is developed on a multi-agent platform to provide autonomy and co-operation.

In our view, a co-operative autonomous decision-making system must at least satisfy the autonomy, co-operation and intelligence. For the purpose of this research, these can be explained as:

Autonomy: The system must make decisions on the user's behalf; unlike a traditional system where the system needs a command from an input device. More specifically, the autonomous system must have control over its internal state and actions. For example, a traditional decision-making system remains passive to execute specific tasks. However, an autonomous system would execute the assigned tasks on its own for the achievement of predefined goals.

Intelligence: The decision-making system must exhibit intelligent behaviour by applying expert knowledge to manipulate the environment. This would involve recognising the relative importance of different elements in a situation and then responding in a flexible and quick manner to fulfil predefined objectives.

Co-operation: The autonomous and intelligent decision-making tool must also demonstrate co-operative behaviour. This means that the individual agents within the system would work to achieve the overall goal of the system without conflicting with each other. This builds upon the concept of socialability, i.e. the agents would autonomously communicate and negotiate with each other. We realised that the traditional technologies lack autonomy and co-operation; it was decided to propose a system by combining fuzzy logic and multi-agent technologies. Fuzzy logic was chosen to build our system mainly because according to Lotfi Zadeh cited by Steimann [1, 2]

Indeed, the complexity of biological systems may force us to alter in radical ways our traditional approaches to the analysis of such systems. Thus, we may have to accept as unavoidable a substantial degree of fuzziness in the description of the behaviour of biological systems as well in their characterisation. This fuzziness, distasteful though it may be, is the price we have to pay for the ineffectiveness of precise mathematical techniques in dealing with systems comprising a very large number of interacting elements or involving a large number of variables in their decision trees.

Although many intelligent systems have incorporated the learning aspect, and are thus learning-based systems, in our view, the system we are proposing does not need

to have learning capabilities if expert knowledge is already built into the system. The idea behind this argument is that the expert will monitor the behaviour of the system on a regular basis and tune the fuzzy rules if necessary, rather than the system learning on its own in this highly complex domain.

Figure 1 below is an overview of the system.

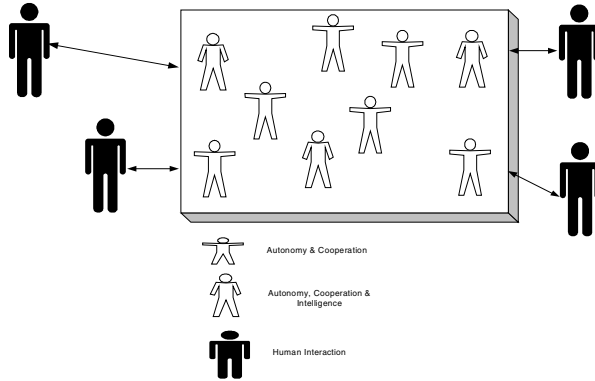


Fig. 1. System overview

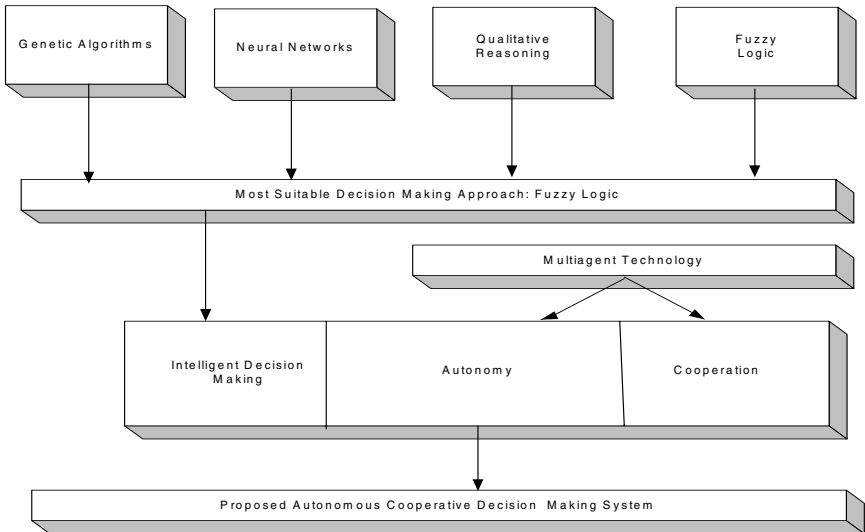


Fig. 2. Autonomous co-operative decision-making system

The aim was to ensure that Fuzzy Logic and multi-agents system complement each other and after reviewing different decision-making technologies we came up with the model shown in Figure 2. The interesting thing to reiterate at this stage is if we are applying this model in domain where we need a different set of skills i.e. learning behaviour or qualitative reasoning we can chose Genetic Algorithm or Qualitative Reasoning and complement it with multi-agents.

2 The Integrated System

The integrated system would demonstrate: how specialist advice that is underpinned by rules based on the knowledge and expertise of the expert. how complex nature of the interaction in that there are concurrent multiple roles and/or actors that require independent dynamic organizational capability to handle the logistics. how advice is dynamically determined using current data provided by users and the above decision-making technologies to provide individualized and tailored prescription. The system would have to be dynamic, capable of making decisions at run time, not requiring them to be wired in at design time, like traditional distributed systems [8]. They would have to perform independent actions on behalf of their owner or user, and work out how to achieve their design objectives, autonomously and dynamically [8]. This is because they are dealing with current, individual data. Therefore, the advice needs to be personalized according to each specific case. At the same time, it has to be expert advice.

2.1 Assigning Agents

Our focus was on replicating the real world problem as is and mapping it onto the chosen technology. Every agent in the system replicates functions of the real world. For example, the Patient Agent of the Diabetes Advisory System replicates the actions of the patients as they would perform in the absence of this autonomous system. In such a scenario; Patient Agent is signifying and replicating the patient. Requesting advice after giving details about their glucose level, activity and diet, receiving this advice, giving details about their schedule, etc., are all examples of actions that have been simply mapped onto agents. Similar activities were grouped as follows. These concepts are illustrated in Figure 3. Six agents were identified based on similar activities: Patient Agent, Doctor Agent, Expert Agent, Doctor Schedule Agent, Reminder Agent and Stock Agent. The aim was to ensure that an adequate number of agents were being used, such that the objectives of the system were being met, that a particular agent was performing similar type of activities, no activity was being repeated or overlooked and there was clarity and coherence in the system.

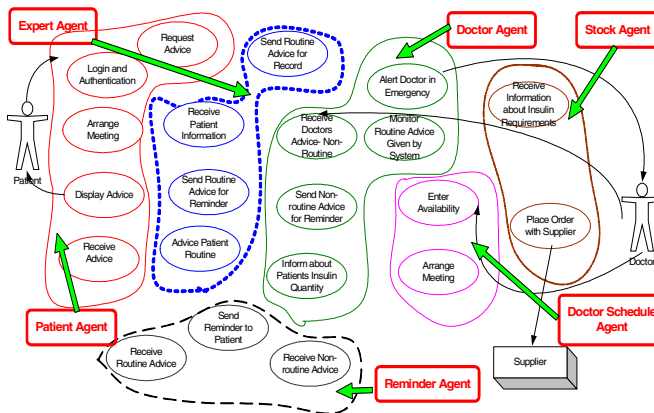


Fig. 3. The diabetes system use case

2.2. System Level Design

At the agent system level design stage, two models are presented. First is the agent system architecture that describes agent interaction diagrams and second, the system plan (sequence diagram) using the Agent-based Unified Modelling Language (AUML) [9] concept. AUML is an extension of Unified Modelling Language (UML) [10], which has already gained wide acceptance for the representation of engineering artefacts. AUML basically synthesises a lot of agent-based methodologies.

2.3 System Plan Model

In the previous section, agents are identified for the system. This section shows the dynamic interaction of agents. This model in Figure 4 is adapted from AUML. The system plan model shows the different roles of agents, how they are interacting with one another and the communicative protocols between them. In order to express in detail the communication between different agents this research has followed the FIPA [11] guidelines. The communication protocols used in this research are: inform, request and propose protocols. For example, in Figure 4, the Patient Agent sends information about the patient’s activity level, diet and glucose level to the Expert Agent, and requests for insulin advice, the communication is two-way and the protocol used is Request [12]. Another protocol used is the Inform protocol [12], for example, when the Expert Agent informs the Reminder Agent about the insulin advice. This is a one-way communication. The nature and sequence of individual interactions is explained in detail in the next section.

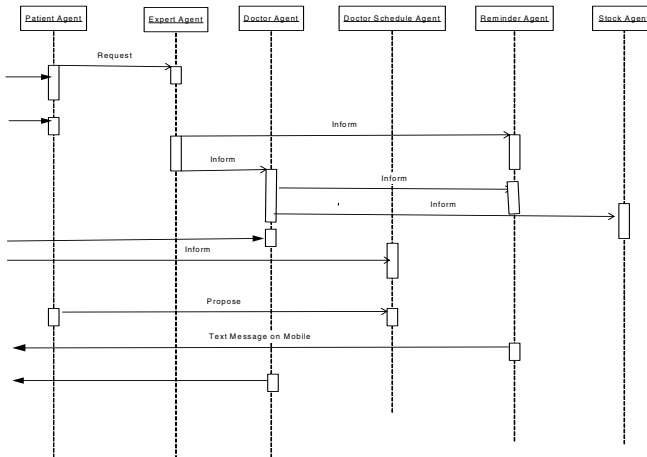


Fig. 4. Agent interaction sequence diagram

2.4 Agent Level Design

This section focuses on the design of individual multi-agent. We will describe the design of one agent. A Class-Responsibility-Collaborator (CRC) index card [13] is used to represent collaboration. CRC cards provide an effective technique for

exploring the possible ways of allocating responsibilities and give high level descriptions of functions to classes and the collaborations that are necessary to fulfil the responsibility [14]. The strength of CRC cards lies in the fact that, when utilised by a team, the interaction between the members can highlight missing or incorrect items in the class. Also, the relationships between classes are clarified when CRC cards are used.

Expert Agent: The Expert Agent has the fuzzy rules that form the expert system. This section is an explanation of the interactions of Expert Agent shown in Figure 4. Table 1 is an Expert Agent CRC card.

Table 1. Expert Agent CRC Card

Expert Agent CRC Card	
Responsibilities	Collaborations
Advise the patient	Patient Agent
Send insulin advice	Doctor Agent
Send insulin advice	Reminder Agent

Table 2. Expert agent collaborations

Agent Name: Expert Agent		
Sending Message to	Receiving from	Message
Patient Agent Message type: <i>Inform</i> Content: <i>Insulin Advice</i>	Patient Agent	Message type: <i>Request</i> Content: <i>Activity level, Glucose, Diet.</i>
Doctor Agent Message Type: <i>Inform</i> Content: <i>Insulin Advice</i>		
Reminder Agent Message Type: <i>Inform</i> Content: <i>Insulin Advice</i>		

Responsibilities

The Expert Agent receives requests for insulin advice from the Patient Agent. It advises the Patient Agent and sends this information to the Reminder Agent as well, so that the Reminder Agent can send a reminder to the patient regarding when to take what quantity of insulin. This information is also sent to the Doctor Agent for its record. The Doctor Agent monitors this value. This action acts as a double check.

Collaborations

As seen in Table 1, the Expert Agent communicates with the Patient Agent, Doctor Agent and Reminder Agent. The collaborations between them are shown in the Table 2 in the form of sending and receiving messages.

3 Development

The focus of this section is on the development phase. As discussed earlier the overall goal is to reduce routine consultations with the doctor. To achieve this goal, it was decided to divide this into four sub-goals so that this task can be achieved easily.

3.1 Agent Plan Model

The agent plan model shows the agent's internal tasks and the data structures used for agent interaction. The goal of the agent plan model is to show the activity of each agent in detail, the state of the agent and its interaction with different agents along-with the type of message. Standard AUML symbols are used. We will discuss the working of one agent, patient agent. The different communications as shown in Figure 5 are:

- C 1: Requests advice for insulin intake from Expert Agent.
- C 2: Throws an exception, ACL (Agent Communication Language) message not understood.
- C 3: Throws an exception, ACL message not understood.
- C 4: Proposes meeting with Doctor Schedule Agent.

Figure 5 is an agent plan model that shows the Patient Agent’s collaborations and interactions with other agents. The initial state of the agent is waiting until a condition is fulfilled. The first condition is, patients enter their ID and password for the authentication process, and then their glucose level, activity and diet for advice from the Expert Agent for insulin. Patients also enter their schedule, which would be used for meeting negotiation.

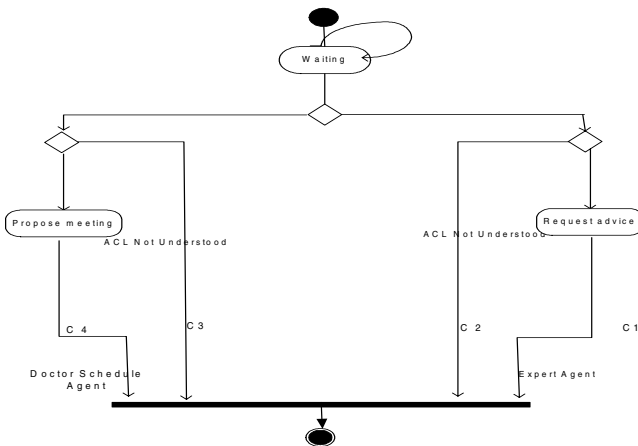


Fig. 5. Patient Agent - Agent Plan Model

3.2 Communicative Act amongst Agents

This section explains how agents are communicating with each other. The agent interactions shown below are based on the Agent Interaction Sequence Diagram (figure 4).

Here, more details are given regarding the interaction between the individual agents as well as an explanation of these interactions. This section is based on the Communication Act Library (CAL) of FIPA . According to FIPA, agent communications have to adhere to a standard procedure, as explained in the following sections.

3.3 Patient Agent and Expert Agent Communication

Figure 6 is a detailed explanation of the interactions shown in Figure 6. According to FIPA protocol, agents have to adhere to a standard procedure. In Figure 6 Patient Agent sends request to Expert Agent, using FIPA *Request* protocol [11] and the contents of the message are Activity level, Glucose and Diet. Once the Expert Agent receives this request it can either refuse it or agree to it. If it refuses, then that is the end of the communication. If it agrees, then, based on the knowledge or fuzzy rules the Expert Agent is linked to, it advises the Patient Agent.

Implementation and evaluation of this system is carried out mainly taking into consideration:

- Ease of use
- Effectiveness of advice

It was also considered that the system developed should be as user friendly as possible, as majority of the patients may not necessarily be IT savvy and this system should complement in easing their lifestyle and also save the doctor's time from regular consultations. This system is developed taking into consideration that diabetic patients can seek medical advice from the comfort of their homes. The importance and reason for envisaging this system has already been discussed in earlier sections.

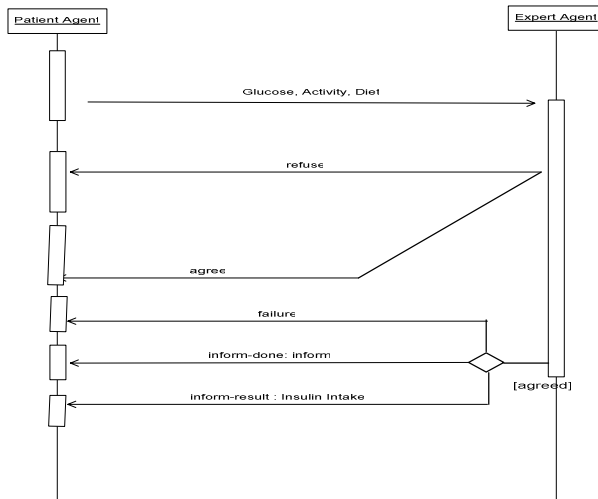


Fig. 6. Patient Agent and Expert Agent Communication Diagram

The patient using the system should have a basic knowledge of diabetes, know how to measure glucose and in which category of diabetes they fall into, as different people have different insulin sensitivities to normalise values that represent the liver. This means that different people respond to insulin in different ways. People with insulin resistance (low insulin sensitivity) require much more insulin to change their blood glucose concentration than people with high insulin sensitivities (low insulin resistance). In other words, with low insulin sensitivity a lot of insulin is required to have a small effect on the blood glucose profile, while with a high insulin sensitivity a small amount of insulin is required to have a large effect on the blood glucose profile. The liver value represents the insulin sensitivity of the liver while the peripheral value represents the insulin sensitivity of the rest of the body.

4 Intelligence in Agents

The intelligent aspect of the proposed system was implemented using a rule-based design, with fuzzy categorisation for both inputs and outputs. A rule-based approach was chosen because it is a tried and tested approach used in the field of medical expert systems [15, 16]. Figure 7 is a fuzzy membership function for blood glucose level. This figure starts from hypo stage or low stage of glucose level where glucose level is very low and the patient is advised to reduce insulin intake and eat something as prescribed by the doctor to attain the minimum level of glucose in his blood.

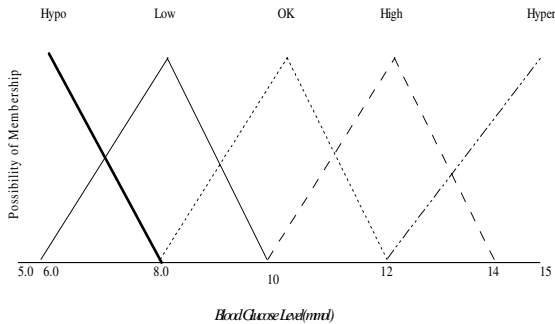


Fig. 7. Membership Function for Blood Glucose

Figure 8 is a fuzzy membership function for activity, it starts from sleep mode where the patient burns the least amount of glucose, to very strenuous, where the patient burns maximum glucose.

Figure 9 is also a fuzzy membership function that recommends the insulin dosage. The fuzzy rules shown in Figure 7, 8 and 9 can further be explained below:

IF the activity is SLEEP and the glucose level is HIGH then INCREASE Insulin.

IF the activity is VERY STRENUOUS and the glucose level is VERY LOW then LARGE DECREASE Insulin.

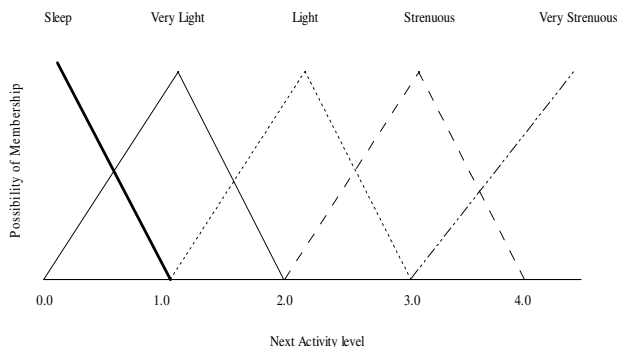


Fig. 8. Membership Function for Activity

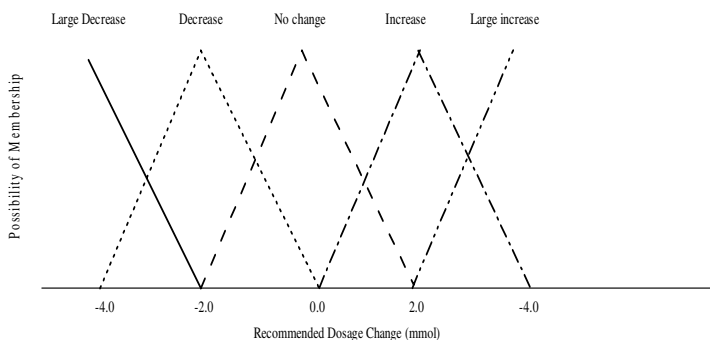


Fig. 9. Membership Function for Recommended Dosage

The next section focuses on the implementation of the system. There are several issues to be considered in order to implement an agent environment, such as agent management, security, services and directory. The agent environment provides the criteria of behaviours, communication, distribution and openness. This comprised of several combinational technologies, such, as concurrency and distribution [17].

5 Conclusion and Discussion

Diabetes, as mentioned in the introduction to this paper, is indeed a very serious problem. It is intended that the system will bring ease into the life of diabetics, at the same time freeing up the resources of the healthcare providers. In other words, the possibility to provide anytime-anywhere diabetic advice would make a diabetic feel at ease as they can always get in touch with their primary health care provider. And the healthcare providers can focus on vital and non-routine tasks.

The system, from a system-design perspective, shows evidence of a flexible design. What this means is that it possesses the elements of scaling up, functioning in a distributed environment, and above all, autonomy. This puts forward a framework where multi-agents and humans can work together to achieve desired goals. The

experience learned from developing this system so far has proved the concept that the fuzzy and multi-agent technologies can provide diabetes advisory service from a remote location. The steady expansion of the internet and the work on platform-independent software is making the remote service implementation faster and more cost-effective. It can also be argued that because of the nature of the proposal, this framework can be adapted to other clinical advisory problems.

This research was undertaken to look for directions to improve decision-making in healthcare systems. The idea was to build on the benefits of traditional decision-making technologies and complement them with other new technologies. Therefore, the aim was to exploit the advantages of different domains. This study, like every other study, is limited by its scope. The scope of this study was to present a framework to improve decision-making in the healthcare domain. This did not include implementation and evaluation of this framework in the real world. In order to remain focused on our objective, we have restricted the research to construction of the framework and envisaging the advantages that would accrue on its implementation.

The argument goes that delegating tasks to computers is a compromise on quality of service of healthcare service providers. And there is a trade-off when we are talking about delegating even routine tasks to computers. But in reality, with the limited resources complemented with an ageing population we need a paradigm shift, which can be achieved by meeting stringent requirements. As mentioned earlier, the main goal of majority of the works reported here was to optimise the cost of care in the healthcare domain by empowering a person to use the expert's knowledge captured in the decision-making tool. In our case, this person was the patient whereas, in the majority of these works, this person was the doctor.

The proposed work accordingly involves not just the understanding of computer technology but also the social and behavioural impact this would have on the overall process. Another important aspect is, how new technology is introduced in the establishment. This task becomes even more daunting when it involves the healthcare domain because of obvious reasons. Evaluation in the real world also depends on other external factors such as: legislation, economic constraints, social and ethical aspects, for example. However, the focus of our work was on impact and outcome and benefits these systems can bring about rather than looking into all the aspects we have briefly touched above. And in reality these issues would have to be dealt with on a case-by-case basis and the developed system may not offer much help. Nevertheless, the developed system would provide a framework in which these issues and questions can be identified and articulated more easily.

References

1. Phuong, N.H., Kreinovich, V.: Fuzzy logic and its applications in medicine. *International journal of medical informatics* 62(2-3), 165–173 (2001)
2. Steimann, F.: Fuzzy Set Theory in Medicine. *Artificial Intelligence in Medicine* 11, 1–7 (1997)
3. IDF: Diabetes On The Rise World-Wide, <http://www.docguide.com>
4. Diabetes Basics - What is Diabetes?, <http://www.lifeclinic.com>
5. Daly, H., et al.: *Diabetes Care: A problem Solving Approach*. Heinemann Professional Publishing (1984)

6. Hill, R.D.: Diabetes Health Care. Chapman and Hall, London (1987)
7. Lasker, R.D.: The diabetes control and complications trial. Implications for policy and practice. *New England Journal of Medicine* 329, 977–986 (1993)
8. Wooldridge, M.: An Introduction to MultiAgent Systems, p. 348. John Wiley & Sons, Ltd., Chichester (2002)
9. Odell, J., Parunak, D.V.H., Bauer, B.: Representing Agent Interaction Protocols in UML. *Agent-Oriented Software Engineering*, pp. 121–140 (2001)
10. Object Management, G, Unified Modelling Language Specification Version 1.3, in OMG Document (1999)
11. FIPA Communicative Act Library Specification, <http://www.fipa.org>
12. FIPA Request Interaction Protocol Specification, <http://www.fipa.org>
13. Wirfs-Brock, R., Wilkinson, B., Wiener, L.: Designing object-oriented software, p. 341. Prentice-Hall, Englewood Cliffs (1990)
14. Bennett, S., McRobb, S., Farmer, R.: Object-Oriented Systems Analysis and Design using UML. McGraw Hill, New York (1999)
15. Frenste, J.H.: Expert Systems and Open Systems. In: Proc. AM Assoc. Medical Systems and Informatics
16. Pham, T.T., Chen, G.: Some Applications of Fuzzy Logic in Rule-Based Expert Systems. *Expert Systems* 19(4), 208–223 (2002)
17. Bradshaw, J.M.: Software agents, p. 480. MIT Press, Cambridge (1997)

Towards Virtual Epidemiology: An Agent-Based Approach to the Modeling of H5N1 Propagation and Persistence in North-Vietnam

Edouard Amouroux^{1,2}, Stéphanie Desvaux^{3,4}, and Alexis Drogoul^{1,2}

¹ IRD UR079 GEODES, 32 Avenue Henri Varagnat, Bondy - France

² Equipe MSI, IFI, 42 Ta Quang Buu, Ha Noi - Viet Nam

³ CIRAD, Campus International de Baillarguet F-34398 Montpellier - France

⁴ PRISE Consortium in Vietnam c/° NIVR, 86 Truong Chinh, Ha Noi - Viet Nam
{edouard.amouroux, alexis.drogoul}@ird.fr,
stephanie.desvaux@cirad.fr

Abstract. In this paper we claim that a combination of an agent-based model and a SIG-based environmental model can act as a “virtual laboratory” for epidemiology. Following the needs expressed by epidemiologists studying micro-scale dynamics of avian influenza in Vietnam, and after a review of the epidemiological models proposed so far, we present our model, built on top of the GAMA platform, and explain how it can be adapted to the epidemiologists’ requirements. One notable contribution of this work is to treat the environment, together with the social structure and the animals’ behaviors, as a first-class citizen in the model, allowing epidemiologists to consider heterogeneous micro and macro factors in their exploration of the causes of the epidemic.

Keywords: Multi-Agent Systems, Agent-Based Models, epidemiological models, environmental models, GAMA platform.

1 Introduction

Over the past few years, avian influenza spread from Asia to Europe and parts of Africa. Following this proliferation, a certain downturn has been observed thanks to concerted measures (improving hygienic practices, vaccination programs, etc). Yet, total eradication remains elusive. This disease remains a major threat to both the economy and public health. Within this context, the challenge for current epidemiology is to eradicate the virus, which requires understanding the various factors that may impact the proliferation of this infection. In particular, the local propagation of the virus and its re-emerging mechanisms are yet to be fully understood. Current hypotheses point an accusing finger towards: the presence of wild birds traditional farming practices and trading activities. None of these have been validated as of yet. Recent studies [1] suggest that human activities, particularly within the agricultural system, in dynamic interaction with the environment, play a key role in the spread of the disease at local levels (within a given district). To have a better understanding of these mechanisms, one of the possible methods is for epidemiologists to consider

simulated models of reality and to test hypotheses concerning the environment, the social structure, the behaviors of the birds, etc. Unfortunately, existing modeling techniques have not reached a level of maturity appropriate for such an exploratory use of simulation. The aim of this paper is to present an Agent-Based Modeling (ABM) approach, while detailing how it can be used to help epidemiologists answer their questions. After presenting the context and requirements of this work, we will review existing epidemiological models, explain why they do not fit the requirements and why we have chosen to develop our own tools on top of a multi-agent platform.

2 The Context of Avian Influenza in Vietnam

Avian influenza (HPAI) epidemics occurred recurrently in Vietnam since mid-2003. While the outbreaks in both North and South Vietnam are similar the underlying variables of their respective poultry production industries are different: climactic and environmental conditions, variety of circulating virus strain [2] and poultry production organization [3] are distinctive in each region.

2.1 Two Epidemiological Questions to Address

As of today, many basic epidemiologic questions (about avian influenza) still need to be answered. At a macro level, propagation is not precisely understood but the general tendencies are. At the micro scale (e.g. village or commune level), the explanations of propagation mechanisms are hazy at best. Understanding what is happening at these micro-levels is probably the key to controlling the epidemic. Hence many investigations are focusing on them, particularly on these key questions:

- (1) **Propagation mechanisms:** how does the virus propagate locally (from birds-to-birds, humans, environments, farms, markets, etc.)?
- (2) **Persistence mechanisms:** how can the virus re-emerge in a previously uninfected area, sometimes months after the end of the previous wave of the epidemic?

2.1.1 Local Propagation Mechanisms

Both mechanisms can be explained by the presence of wild birds [4] and by trading activities. Other recent hypotheses concern the role of the agro-system. Recent policies focus on small farmers for their lack of biosecurity. Studies are also concentrating on semi-industrialized farm—as they have a much wider span of influence—yet biosecurity is low. As fully industrialized farms have adopted modern, state-of-the-art biosecurity systems, they present a lesser concern for the experts. Whatever the considered production sector is, the entire agro-system (the farm, suppliers and customers) is being monitored extensively by epidemiologists.

2.1.2 Re-emergence, or Persistence Mechanisms

Re-emergence is another concern. The prevailing hypothesis is that wild birds act as reservoirs but validation is lacking at present [4]. Another hypothesis concerns the environment itself as a reservoir. In-vitro experiments are currently being carried out on the persistence of H5N1 within the environment and especially in water [5]. It has been demonstrated that avian influenza viruses are likely to remain infective several months while in places like ponds [5].

2.2 Addressing Epidemiological Questions through Simulated Experiments

2.2.1 Towards Simulated Experiments

Local-scale propagation, persistence and re-emergence processes are a complex matter where many actors (birds, domestic or not, environment, human activity, etc...) intervene. Unfortunately, the data that has been produced to date is insufficient in term of completeness and reliability. This situation will probably remain unchanged in the near future. To have more reliable data, in-vitro studies are being conducted, but the conditions are too far from reality to be easily transposed to field reality. Since neither field nor in-vitro studies can be completely satisfactory, experts turn towards models and simulation experiments to validate their hypotheses, and explore new ones.

2.2.2 Requirements for "Virtual Epidemiology" Experiments

As we will see in the next section, existing modeling techniques are not really suited to such "in silico" experiments. Indeed, the ideal tool would need to respond to several requirements expressed by epidemiologists. The most important are listed below:

- (1) The *environment* should be represented *extensively* together with its *own dynamics*. This representation should also enable the use of various and heterogeneous descriptions (geographical, social, ecological, etc).
- (2) Epidemiologists need the ability to work both at the *population level* and directly on models of *individuals*, depending on the hypotheses requiring validation.
- (3) *Interactions at the population or individual levels*, but also *with the environment*, should be modeled explicitly, since a change in these interactions may have an impact on the dynamics of the epidemic.
- (4) These data are usually expressed using *heterogeneous formalisms* that the expert needs *to be able to reuse* in the modeling of the system.
- (5) The simulation platform should be able to recreate an environment in which "*virtual experiments*" could be run (exploration of parameters, etc.)

3 Epidemiological Models: A Brief Review

In this section, we review the existing epidemiologic models in the light of these requirements. Although the literature on the subject is quite important, we will see that most of the offers do not fulfill them completely yet.

Table 1. Summary of the relationships between questions and models

How to ...	Model	Referece
Represent and study epidemics at a global level?	SIR Bayesian Networks HMC	Yorke 79 Abbas 04 Durand 99
Account for the heterogeneity of the population?	Micro-simulation	Artzrouni 01
Represent contact patterns?	Cellular automata Social networks	Turner 01 Ghani 07
Represent different individual behaviors?	Individual based	Wilensky 98
Study the role of the environment?	Agent based Agent based + GIS	Muller 04 Badariotti 05
Enhance the representation of the environment?	Agent based + GIS	Badariotti 05

Although many types of model have been proposed and used for epidemiological ends, every one of them still lacks the ability to address some of the requirements expressed by epidemiologists in this type of research (see 2.2.2).

It is plain that global models such as SIR [6], bayesian networks [7] or hidden Markov Chain [8] will not work when considering small-scale situations. Micro-simulation, [9] & [10] will work, but lacks the ability to take situatedness into account. This can be addressed by cellular automata [11] or social networks [12], depending on the level of complexity to be modeled. These models cannot represent fully heterogeneous agents (internal state + behaviors)—while ABM [13] does bring this capacity to the table. Finally, this kind of model generally lacks a detailed expression of the environment—among our top concerns in this application. To address this issue, computer scientists, [14] & [15], have proposed the joint use of ABM and GIS but they have not yet fully implemented this idea.

Thus, the natural choice for us, was to follow an agent-based approach, coupled with a detailed and flexible representation of the environment based on a combination of grids and GIS. Additionally, we decided to make the environment (and its components) a “first-class citizen” of the model, provided with its own attributes and behaviors, rather than just considering it as a topological surface. This extension to “classical ABMs” will be described in the next section through the presentation of the HPAI in North Vietnam model built in cooperation with epidemiologists.

4 Conceptual Model of Avian Influenza Propagation

In this section, we introduce the context of this research, followed by the representation of the environment and actors. All the choices made in accordance with epidemiologists and field specialists will be explained.

4.1 Frame of the Epidemiologic Study

The study takes place in North Vietnam, where epidemiologists focus on “local” mechanisms, i.e. mechanisms that occur at a scale comprised between the village and the district levels (around 50 km²). The model is then geographically limited by the bounds of a province (a few hundreds km²). The main geographical entity is the village, which is considered by epidemiologists as a “coherent epidemiologic unit”. Their hypothesis, present in the model, is that communes and districts are not really relevant to consider when it comes to studying the local causes of propagation. This assumption may however be easily revised in future occurrences of the model. As the environment may be a reservoir for the virus, we will consider that every place may allow virus survival. In addition all entities in the model can be also infective.

The “village” environmental unit

A “traditional” Vietnamese village consists of an inner-village space with a main street, a few dozen to more than a hundred households with some poultry (traditional farms) and pets. This inner space is surrounded by rice fields, watered lands (which enter the village), other cultures and is protected from the flooding by a dike. The organization of the village is described in [16] while statistic will be provided by the Vietnamese census and longitudinal surveys currently conducted.

The environment

The surroundings of the village consist of agricultural and “natural” lands (river, forest, etc.). As we do not focus on long distance propagation, we consider this distant space as homogeneous. Conversely, the inner-village and agricultural lands are represented in detail, especially the possibility that the latter may act as a reservoir for the virus. According to epidemiologists, viral dynamics within the environment are susceptible to: *altitude, Ph, temperature, solar exposure, level of organic matter* present in water, *watered or not*. These parameters define the ecological dynamics of the system. These parameters vary greatly in natural or cultured land we can consider them to be static in the inner village and roads.

4.2 Relevant Actors

Actors of the pathogen system act at different scales and are organized in 2 main structures: the village and the poultry production chain. The other levels of organization (like administrative levels) are neglected in the model, as they do not impact local scale dynamics.

Actors of the Traditional Village

In the village important actors are organized around the farm. The type and dynamics of both are defined according to the production type and the production sector (see below).

Farm and markets

We can classify farms by their size and production techniques in four sectors according to [17]:

- *Traditional farming*: mixed poultry, local scale interactions, no bio-security at all.
- *Semi-industrial & industrial*: targeted production, district to province size scale interactions, low/medium bio-security level (medium/high for industrial farms).
- *Fully integrated*: targeted production, province size scale interactions, sophisticated bio-security systems (thus, not considered in the model).

The production type and sector determine most of the farm’s characteristics in terms of *herd (size, lots organization, vaccination coverage, species and breeds)*, the type of premises (caged, fully confined, pond), which impacts the *bio-security level*, etc. They also determine the dynamics of the farm and its acquaintances (processing chain).

Markets are also a key location within the village. It acts both as a reservoir for the virus and an exchange place, as there is extensive contact between processed and living poultry. The latter can be brought back home newly infected if not sold.

Poultry

According to experts, all the poultry within a flock are very similar in terms of behaviors (gregarious animals), and characteristics (homogeneous lots). So we can aggregate these individuals and represent only the poultry flock. This, plus homogeneous mixing occurring within the flock allow us to represent the virus transmission within the flock with a SIR model. A traditional farm’s flock will be addressed by developing an adapted SIR model as they are not homogeneous lots. The flock behavior depends on the farm type and sector, the poultry can be confined, access an adjoining or distant secured zone (a pond, channel, etc), or be free ranged.

Humans

Humans may play the role of mechanical vector in local propagations. However, their impact is considered as very low when compared to poultry flocks. Consequently, we neglect them in this first model. (Para-) veterinarians are a different case, as they travel among numerous farms and are not always fully trained they may impact the disease dynamic, thus they will be represented explicitly, going farms to farms.

Animals

Wild birds, which are considered by most epidemiologic surveys as not playing a noteworthy role in local propagation, are not to be considered. Peridomestic birds, farm animals, pets and fighting cocks are also removed from this model, as the data available on them regarding their role in avian influenza is not clear though surveys are being conducted and their integration is planned.

4.2.1 Outside the Traditional Village

Villages are also linked to poultry production networks, which have a much wider scale dynamic and a possible strong impact because they connect all the farms and may constitute a good propagation system.

Mainly, we can consider ([3] and expert knowledge) four actors of this production chain: traders (highly variable in term of size), transporters, slaughterhouses and selling points. Traders can stock, trade among themselves and manage their transporters, who carry living or slaughtered poultry from one chain production node to another.

Slaughterhouses “just” process the poultry and thus have no relevant dynamic. The only point of sale of processed poultry (i.e. supermarket) can be neglected as it is considered as a disease end point (no infection possible). On the contrary, local, district or provincial markets need to be represented in the same way as the commune.

We have a fairly complex system with both a detailed environment and numerous actors to represent—but, at this point, the general structure has been defined. The environment is split into three main categories: a static inner village, the agricultural surroundings (which is dynamic) and the far surrounding areas (considered static and homogeneous). Actors are village-related and connecting villages are related.

As all needed data are not yet available, the creation process of the model will be incremental. We are conscious that we must remain prepared to integrate any entities that may be declared relevant from surveys being conducted presently.

4.3 Implementation

To implement this model we use a versatile ABM simulation platform, GAMA [18]. The specific feature of this platform is that it allows modelers to work with an artificial data environment (a grid, for instance) in the same way as field data environments (i.e GIS) and provide for the seamless integration of both if needed.

4.3.1 The Environment

Similarly to [15] we use a mix of grids, unavailable data and within-simulation generated data, and GIS objects representing field data information. Both contain the environmental data as described in the previous section. These parameters are used to define the endogenous dynamic of the environment, and the virus’ evolution in

particular. This allows epidemiologists to work on real data (GIS) and incomplete data but also to infer unavailable data and test the plausibility of such data.

4.3.2 Agents

Classically, we use reactive agents but they are situated in a complex environment (GIS & grids) and they have also persistent action (for example the daily behavior of a poultry flock). The modeling of poultry is of particular interest. A flock is homogeneous in term of structure and behaviors thus it can be represented with one agent representing the group. Although, they differ in regards to virologic history, as they are not infected all at the same time and respond differently to the disease. Consequently, we added a matrix of individuals in each flock as an agent, in the manner of micro simulation.

5 Conclusion

In this paper we have presented a brief review of epidemiologic questions and the models adept at responding to those questions. We moved away from the field of mathematics, while addressing global-scale questions, using computer-based models and especially IBM which are better suited for a small-scale problematic. Afterwards, we presented a model to study small-scale epidemiological phenomenon in the context of avian influenza in North Vietnam: local propagation and re-emergence mechanisms. This model differs from those previously proposed, as it treats the environment as a first-class citizen of the system. Here the environment is not only a topological surface, it can contain heterogeneous data, have its own dynamic and can be multiple (GIS + grid in our application). In future works, validation is our most important concern. Although this model is explanatory and exploratory rather than predictive in nature, we consider validating the predictive capabilities of the model paramount to our future work.

References

1. Gilbert, M., Xiao, X., Pfeiffer, D.U., Epprecht, M., Boles, et al.: Mapping H5N1 highly pathogenic avian influenza risk in Southeast Asia. *Proc. Natl. Acad. Sci. USA.* 105, 4769–4774
2. Dung Nguyen, T., Vinh Nguyen, T., Vijaykrishna, D., Webster, R.G., Guan, Y., Peiris, J.S.M., Smith, G.J.D.: Multiple sublineages of influenza A virus (H5N1), Vietnam, 2005–2007. *Emerging Infectious Diseases*, 14, 4 (2008)
3. Agrifood consulting international: *The Impact of Avian Influenza on Poultry Sector Restructuring and its Socio-economic Effects* (2006)
4. Chen, H., Li, Y., Li, Z., Shi, J., Shinya, K., Deng, G., et al.: Properties and dissemination of H5N1 Viruses isolated during an influenza outbreak in migratory waterfowl in Western China. *J. of Viro.* 80, 5976–5983 (2006)
5. Ito, T., Okazaki, K., Kawaoka, Y., Takada, A., Webster, R.G., Kida, H.: Perpetuation of influenza A viruses in Alaskan waterfowl reservoirs. *Arch. of Viro.* 140(7), 1163–1172 (1995)

6. McKendrick, A.G.: Applications of mathematics to medical problems. Proc. of the Edinburgh Mathematical Society 44, 1–34 (1925)
7. Abbas, K., Mikler, A., Ramezani, A., Meneze, S.: Computational epidemiology: Bayesian disease surveillance. In: Proc. of ICBA 2004 proceedings (2004)
8. Durand, B., Mahul, O.: An extended state-transition model for foot-and-mouth disease epidemics in France. Preventive Veterinary Medicine 47, 1–2 (2000)
9. Artzrouni, M., Gouteux, J.P., et al.: Population dynamics of sleeping sickness: A micro simulation. Simulation & Gaming 32(2), 215–227 (2001)
10. Brouwers, L.: MicroPox: a Large-scale and Spatially Explicit Microsimulation Model for Smallpox Transmission. In: Proc. of the Intl. Conf. of Health Sciences Simulation (2005)
11. Turner, J., Begon, M., Bowers, R.: Modelling pathogen transmission: the interrelationship between local and global approaches. Proc. Biol. Sci. 270(1510), 105–112 (2003)
12. Ghani, A.: Models for an Outbreak of Avian Influenza in the UK Poultry Flock, Journées de modélisation en épidémiologie animale du CIRAD (2007)
13. Wilensky, U.: NetLogo Virus model (1998), <http://ccl.northwestern.edu/netlogo/models/Virus>
14. Muller, G., Grébaud, P., Gouteux, J.P.: An agent-based model of sleeping sickness: simulation trials of a forest focus in southern Cameroon, Éd. Sc. et Méd. Elsevier, Amsterdam (2004)
15. Badariotti, D., Banos, A., Laperrière, V.: Vers une approche individu-centrée pour modéliser et simuler l'expression spatiale d'une maladie transmissible: la peste à Madagascar, cyberge (2007), <http://www.cyberge.eu/index9052.html>
16. Bénédicte, S.J.: Atlas Bac Hung Hai, Gret (personnal communication) (1999)
17. Desvaux, S., Vu, D.-T.: A general review and description of the poultry production in Vietnam. Agricultural publishing house (2008)
18. Amouroux, E., Chu, T.Q., Boucher, A., Drogoul, A.: GAMA: an environment for implementing and running spatially explicit multi-agent simulations. LNCS (LNAI). Springer, Heidelberg (2007)

Measurement of Underlying Cooperation in Multiagent Reinforcement Learning

Sachiyo Arai, Yoshihisa Ishigaki, and Hironori Hirata

Graduate School of Engineering, Chiba University,
1-33 Yayoi-cho, Inage-ku,
Chiba Japan
arai@tu.chiba-u.ac.jp

<http://nexus-lab.tu.chiba-u.ac.jp/sarai/>

Abstract. Although a large number of algorithms have been proposed for generating cooperative behaviors, the question of how to evaluate mutual benefit among them is still open. This study provides a measure for cooperation degree among the reinforcement learning agents. By means of our proposed measure, that is based on information theory, the degree of interaction among agents can be evaluated from the viewpoint of information sharing. Here, we show the availability of this measure through some experiments on “*pursuit game*”, and evaluate the degree of cooperation among hunters and prey.

1 Introduction

In general, interaction among agents is classified as either conflict or cooperation by examining of multiagent’s observable behaviors. Whether conflict or cooperation has been decided by the utility of each agent’s viewpoint, such as a payoff matrix in the game theoretic approach. However, since achieving a goal usually requires a sequence of actions, it is hard to evaluate the utility of each action per time step, separately. Consequently, to see whether there is cooperation or not, we have usually evaluated the number of steps to achieving a goal as the efficiency of their performance. Because if the task needs cooperation, it will be achieved faster with the cooperation than the case without it. Otherwise we have no alternative but to observe and evaluate emerged behaviors of the agents’ qualitatively.

Thus there is no quantitative expression of what is going on among the agents in each time step, and then we just have to observe their behaviors carefully. From an engineering viewpoint, clarifying the process of learning with a quantitative measure is important to improve cooperative task of agents’.

In the following section, we describe our problem domain and related study on the “*pursuit game*”. The measure of interaction is defined and applied to the problem in section 3. In section 4, we show some experimental results, and discuss validity of our measure to see their interaction during the learning period. Finally in section 5, we conclude our research here and describe the future work.

2 What’s a Cooperative Behavior?

2.1 Problem Domain

In this paper, we consider the “*pursuit game*” which has been often studied by many multiagent researchers [2]. Reinforcement learning approach to the “*pursuit game*” has been taken by Tan [5] where Q-learning is applied to make each hunter behave appropriately, and they evaluate several configuration of information sharing patterns. In these works, the amount of communication has been evaluated in terms of the efficiency of agents’ performance, such as number of steps to reach the goal, that would not reflect a certain interaction among the agents. It still remains whether the effectiveness of information sharing to the each agent’s viewpoint.

Related Works: There exist some researches to evaluate global behavior of agents’ by applying Shannon’s measure, *information entropy*. Balch [1] introduced the concept of *team diversity* to evaluate robot societal diversity that is determining whether agents are alike or unlike. Also Parunak and Brueckner [7] are based on information theory to analyze the process of reinforcement learning agent. Although these researches introduced Shannon’s Theorems to evaluate behaviors after learning, they did not focus on the uncertainty of each agent during the learning process where it will be found some interesting properties of interaction among the agents.

2.2 Definition of Measure

The good cooperation will make up for information uncertainty of each agent around its environment, and boost efficiency of their task achievement. To quantify the cooperation, we introduce Shannon’s information theory that focuses on uncertainty in a communication system, that includes several agents, exactly the same as the multiagent system.

Agent Model: An agent is modeled as a reinforcement learning entity engaged in an episodic task in an unknown environment. Suppose an environment is defined by a finite set of state, $\mathcal{S} = \{s_1, \dots, s_h, \dots, s_o\}$, each agent has a finite set of available actions, $\mathcal{A} = \{a_1, \dots, a_i, \dots, a_n\}$, a policy of each agent is described in terms of $\pi(\mathcal{S}, \mathcal{A})$ as shown in Eq. (1). If a single action is strongly reinforced and it is only effective one in the state, we call this state has a deterministic policy, and entropy of this state becomes 0 under our definition. Then, $H(\pi(s_h, \mathcal{A}))$, the entropy of its policy $\pi(s_h, a_i)$ can be defined by Eq. (2).

$$\sum_{i=1}^n \pi(s_h, a_i) = 1 \quad (1)$$

$$H(\pi(s_h, \mathcal{A})) = - \sum_{i=1}^n \pi(s_h, a_i) \log \pi(s_h, a_i) \quad (2)$$

Here, we should consider that an agent does not visit each state of the environment equally. Thus, we take weighted average of $H(\pi(s_h, \mathcal{A}))$ by using the occurrence probability of each state $P(s_h)$. We introduce occurrence probabilities of each state as shown in Eq.(3) and Eq.(4), and define weighted average of $H(\pi(s_h, \mathcal{A}))$ as shown in Eq.(5). In the following sections, we abbreviate H_{normal} to H .

$$\mathcal{S} = \left\{ \begin{array}{c} s_1, \dots, s_h, \dots, s_o \\ P(s_1), \dots, P(s_h), \dots, P(s_o) \end{array} \right\} \quad (3)$$

$$0 \leq P(s_h) \leq 1 (h = 1, \dots, o), \sum_{h=1}^o P(s_h) = 1 \quad (4)$$

$$\begin{aligned} H_{\text{normal}}(\pi(\mathcal{S}, \mathcal{A})) &= \sum_{h=1}^o P(s_h) H(\pi(s_h, \mathcal{A})) \\ &= - \sum_{i=1}^n \sum_{h=1}^o P(s_h) \pi(s_h, a_i) \log \pi(s_h, a_i) \end{aligned} \quad (5)$$

2.3 Mutual Information Measure of Two Agents

Suppose that the environment includes two learning agents A and B . The agents take turns at sensing its environment and taking its action, like $ABAB \dots$. The action set of A and policy of A are represented by $\mathcal{A}_A = \{a_1, \dots, a_i, \dots, a_n\}$, and $\pi_A(\mathcal{S}, \mathcal{A}_A)$, respectively. Similarly, we describe action set of B and policy of B are represented by $\mathcal{A}_B = \{b_1, \dots, b_j, \dots, b_m\}$, and $\pi_B(\mathcal{S}, \mathcal{A}_B)$, respectively.

Under the situation, where more than two agents learn concurrently, the destination state alters during learning process. Because there are multiple agents who learn and decide action independently, the state transition probabilities fluctuate.

As shown in Eq(6) and (7), we define the action selection probability of A 's when A senses $s \in \mathcal{S}$. After A sensing state s , A acts a_i , then destination state from s is represented by s^{a_i} . Subsequently, after A 's action, B senses $s^{a_i} \in \mathcal{S}$, and action selection probability of B 's is described B^{a_i} as shown in Eq.(8) and Eq.(9).

In the case where B has no information about A 's action, B 's action selection probability can be defined as Eq.(10) ~Eq.(11).

$$A = \left\{ \begin{array}{c} a_1, \dots, a_i, \dots, a_n \\ \pi_A(s, a_1), \dots, \pi_A(s, a_i), \dots, \pi_A(s, a_n) \end{array} \right\} = \left\{ \begin{array}{c} a_1, \dots, a_i, \dots, a_n \\ P(a_1), \dots, P(a_i), \dots, P(a_n) \end{array} \right\} \quad (6)$$

$$0 \leq P(a_i) \leq 1, \quad \sum_{i=1}^n P(a_i) = 1 \quad (7)$$

$$B^{a_i} = \left\{ \begin{array}{c} b_1, \dots, b_j, \dots, b_m \\ \pi_B(s^{a_i}, b_1), \dots, \pi_B(s^{a_i}, b_j), \dots, \pi_B(s^{a_i}, b_m) \end{array} \right\} = \left\{ \begin{array}{c} b_1, \dots, b_j, \dots, b_m \\ P^{a_i}(b_1), \dots, P^{a_i}(b_j), \dots, P^{a_i}(b_m) \end{array} \right\} \quad (8)$$

$$0 \leq P^{a_i}(b_j) \leq 1, \quad \sum_{j=1}^m P^{a_i}(b_j) = 1 \quad (9)$$

$$B^A = \left\{ \begin{array}{c} b_1, \dots, b_j, \dots, b_m \\ P^A(b_1), \dots, P^A(b_j), \dots, P^A(b_m) \end{array} \right\}, \quad (10)$$

$$P^A(b_j) = \sum_{i=1}^n P(a_i)P^{a_i}(b_j), \quad 0 \leq P^A(b_j) \leq 1 (j = 1, \dots, m), \quad \sum_{j=1}^m P^A(b_j) = 1 \quad (11)$$

In the following sections, we derive mutual information of A which is defined as Eq.(6) and Eq.(7), and B^A which is defined as Eq.(10) ~Eq.(11). Firstly, we derive a conditional probability and a joint probability of a set of probability event A and B^A as shown in Eq.(12) and Eq.(12).

Secondly, a difference between self-information and conditional information can be derived from $-\log P^A(b_j) - \{-\log P(b_j | a_i)\} = \log \frac{P(b_j | a_i)}{P^A(b_j)}$. Then, it is multiplied by the joint probability, $P(b_j, a_i)$, to take the average of each action of A and B^A , as shown in Eq.(13).

Third, a mutual information $I_s(A; B^A)$ is derived from assigning Eq.(11) (12) to Eq.(13). Then, we take weighted average of $I_s(A; B^A)$ by applying Eq.(3), and finally get the definition of a normalized mutual information $I_{\text{normal}}(A; B^A)$, as shown in Eq.(16). In the following sections, we abbreviate $I_{\text{normal}}(A; B^A)$ to $I(A; B^A)$.

$$P(b_j | a_i) = P^{a_i}(b_j), \quad P(b_j, a_i) = P(a_i)P(b_j | a_i) = P(a_i)P^{a_i}(b_j) \quad (12)$$

$$I_s(A; B^A) = \sum_{i=1}^n \sum_{j=1}^m P(a_i, b_j) \log \frac{P(b_j | a_i)}{P^A(b_j)} \quad (13)$$

$$= \sum_{i=1}^n \sum_{j=1}^m P(a_i)P^{a_i}(b_j) \log \frac{P^{a_i}(b_j)}{\sum_{i=1}^n P(a_i)P^{a_i}(b_j)} \quad (14)$$

$$= \sum_{i=1}^n \sum_{j=1}^m \pi_A(s, a_i) \pi_B(s^{a_i}, b_j) \cdot \log \frac{\pi_B(s^{a_i}, b_j)}{\sum_{i=1}^n \pi_A(s, a_i) \pi_B(s^{a_i}, b_j)} \quad (15)$$

$$I_{\text{normal}}(A; B^A) = \sum_{h=1}^o P(s_h) I_s(A; B^A) \quad (16)$$

The above definition of mutual information can be extended to the case of multiple agents without differentially.

3 Experiments

3.1 Experimental Setting

We consider an $n \times n$ toroidal grid world. A prey, P and two hunters, A and B are initially placed randomly. When prey is blocked by two hunters in a pincer

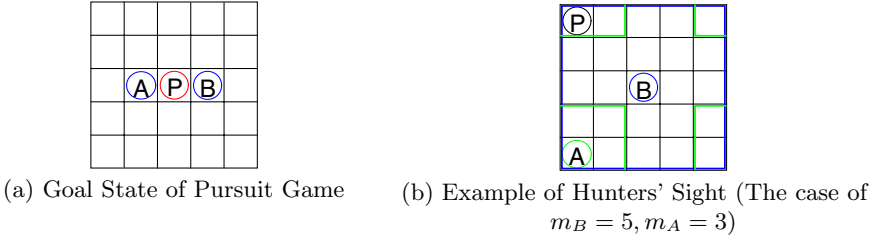


Fig. 1. Experimental Setting of Pursuit problem

movement, as shown in Figure 1(a), before its action, the prey can not move anywhere and then hunters achieved their goal.

Both hunters and prey each successively take turns in moving. Here, it is assumed that there is no communication between agents and they independently sense and act by turns i.e., $ABPABP \dots$. The hunters and prey select the action from the set of $\{Stay, North, East, South, West\}$ but it can not occupy the same position.

Both hunters have a $n \times n$ sight as to the prey, that is prey always comes into hunter's sight. As to the other hunter's location, we describe hunter A's sight for hunter B as $m_A \times m_A$ ($m_A \leq n$), and similarly hunter B's sight for hunter A as $m_B \times m_B$ ($m_B \leq n$). The value of m_A and m_B is changed depending on the purpose of the experiments. In their sight, they can distinguish one from others. Figure 1(b) shows the example of hunter's sight.

A Prey also has a $n \times n$ sight and when a hunter is next to it, it escapes to the other side of the hunter. And if more than two hunters are next to it at the same time, it escapes from the hunter this has the first priority. The order of hunters' priority is *Southern* \succ *Western* \succ *Eastern* \succ *Northern* hunter of the prey. Our setting for the prey's movement is different from the usual setting of "pursuit game" [2][3] where the prey moves randomly anytime. But our setting aims at making it easier to observe and evaluate the degree of emerged cooperation of hunters.

3.2 Learning Algorithm

We use Q-learning [6] that has been often used. It is fact that our proposed measure will be available in the case where agents learn by other reinforcement learning algorithms. To calculate it, the occurrence probability of each state should be given or estimated. The parameters of Q-learning are set to *the learning rate* $\alpha = 0.05$ and *the discounting factor* $\gamma = 0.9$. When the hunter is in the goal state, it receives a reward 1.0. The Q-learning hunter selects its action according to the Boltzmann distribution, $p(a_i|x) = \frac{e^{Q(x,a_i)/T}}{\sum_{k \in actions} e^{Q(x,a_k)/T}}$, where we set $T = 0.5 \cdot 0.998^{episode} + 0.01$. Here, *episode* means the number of steps from the initial state to the goal state. These parameters are chosen by preliminary experiments.

In our experiments, the environment size is $n = 5$, and the sight size of m_A and m_B will be set from $\{1,3,5\}$. In the case of $m_i = 1$, there has no sensory

input of environment, in the case of $m_i = 3$, there happens to be a perceptual aliasing, and in the case of m_5 , an agent can see a whole environment except inside of other agents.

3.3 Experimental Result

We introduce two cases of the experimental results. Both Figure 2(a) and 3(a) show the learning curves where x-axis and y-axis show the number of episodes and the steps of achieving the goal. And (b)-(d) of Figure 2 and 3 show the changing entropy of *Prey's*, the mutual information $I(A; B^A)$, and $I(B; A^{BP})$, respectively. As well, another graph in (a) of Figure 2 and 3 is the magnified one to show their change clearly.

4 Discussion

Agents' Entropy: Figure 2(b) shows that prey's entropy decreases as hunters' learning progressed, though the prey does not learn anything. Since the prey moves depending on the hunters' movement, both hunters make prey not to move randomly for capturing the prey easily. Therefore, the more deterministic the hunters' policies become, the less randomly the prey moves. In other circumstances of the case $m_A = 1, 3$, where the sight size of hunter A becomes

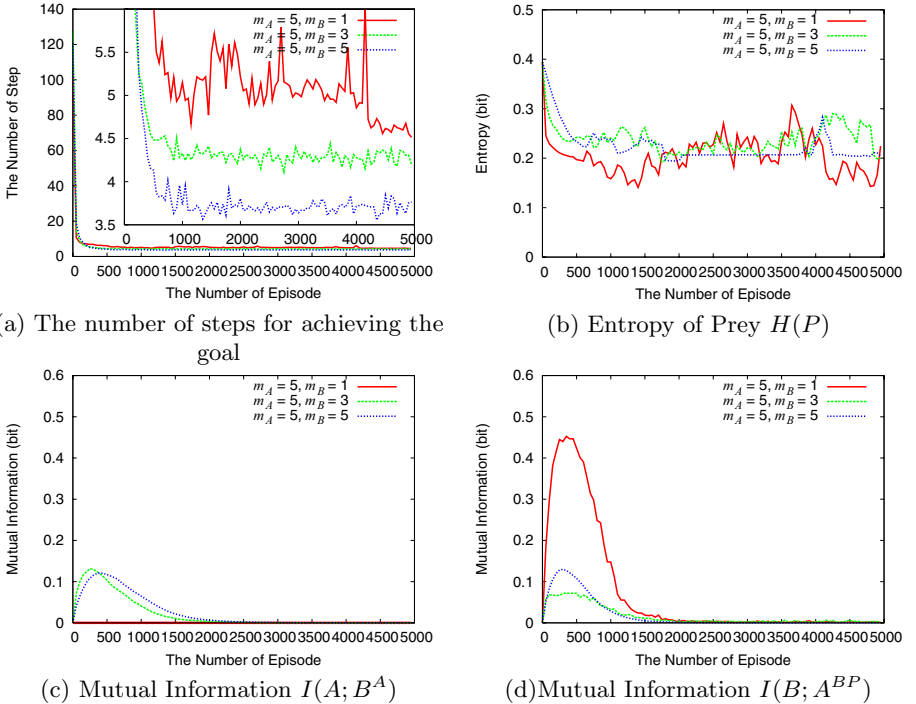
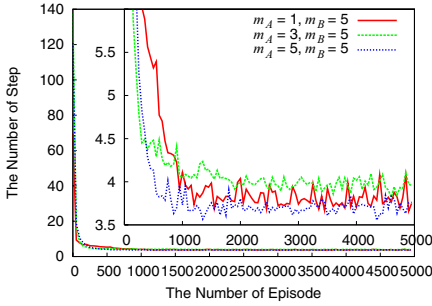
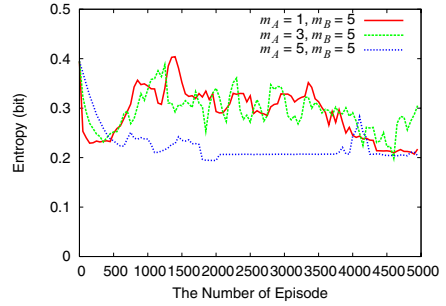


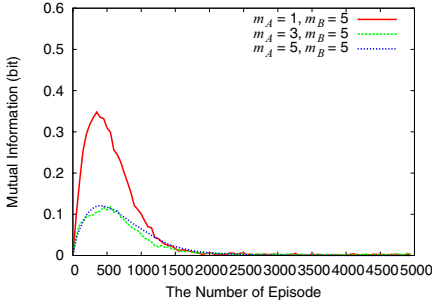
Fig. 2. Result of Case : $m_A = 5, m_B = 1, 3, 5$



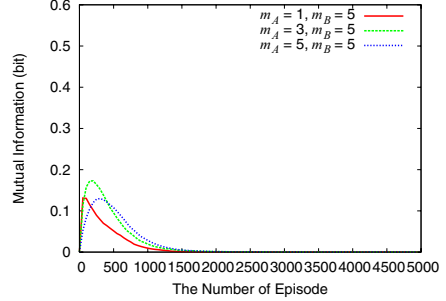
(a) The number of step for solving the task



(b) Entropy of $P H(P)$



(c) Mutual Information $I(A; B^A)$



(d) Mutual Information $I(B; A^{BP})$
($m_A = 1, 3, 5, m_B = 5$)

Fig. 3. Result of Case : $m_A = 1, 3, 5, m_B = 5$

incomplete, the entropy of prey’s does not decrease and it keeps move moving randomly. Keeping prey’s movement random seems effective in this case, because it increases the opportunity to meet the prey.

Mutual Information: Figure 2(c) shows that the value of mutual information $I(A; B^A)$ keeps 0 in the case of $m_B = 1$, where hunter B does not see anything around him. This comes about as a natural result, since hunter B has no option but to act independently from hunter A . On the other hand, we found from Figure 3(d), the mutual information $I(B; A^{BP})$ of case $m_A = 1$, in which hunter A does not have any sight, maintains larger value than that of others. This is due to the order of agents’ action, i.e., $ABCABC \dots$. Since hunter B has a wide view of the environment, it can play effective movement to close the prey. Then the prey tries to escape from hunter B . These interactive actions are very informative for hunter A to move effectively.

Finally, to show the availability of our proposed measure of cooperation, we focused on the mutual information $I(A; B^A)$ of $m_A = 1$ in the Figure 3(c). Here, we found that $I(A; B^A)$ of $m_A = 1$ keeps larger value than the case of $m_A = 3$ and $m_A = 5$. These results indicate that hunter B takes lead role to achieve the goal when hunter A has a smaller sight than hunter B ’s. And vice versa, hunter A takes over the lead, when the site size of hunter B ’s is smaller than that of A ’s, as shown in Figure 2(d). Figure 3(c) indicates another significant aspect of the contribution of our proposed measure. Comparing the value of I in the case

of $m_A = 3$ and $m_A = 5$, there is little difference among them. The result of little difference of I provides quantitative information whether the current sight size is adequate or not. This information is important for modeling agents in the multiagent environment.

5 Conclusion

We introduce information theoretic measure for analyzing emerged behaviors by means of reinforcement learning within the multiple agent environment. Generally, interaction among agents has been discussed qualitatively, that is, there is no option to evaluate whether cooperative behaviors are generated or not, except observational analysis. Whereas observational evaluation should be necessary, it seems not enough to know what is going on in their halfway of learning, or in the process of adaptation to their environment. For this issue, entropy and mutual information that we defined can bring out the interaction among the agents during learning period.

In this paper, we show the availability of our proposed measure through some experimental results on “pursuit game” of which task requires cooperative behavior by nature. We also have found out that our measure gives useful direction even environment where there exist the goal conflicts among the agents. These results will be introduced in another paper.

References

1. Balch, T.: Hierarchic Social Entropy: An Information Theoretic Measure of Robot Group Diversity. *Autonomous Robotics* 8(3), 209–238 (2000)
2. Gasser, L., Rouquette, N., Hill, R.W., Lieb, J.: Representing and Using Organizational Knowledge in Distributed AI Systems. In: Gasser, L., Huhns, M.H. (eds.) *Distributed Artificial Intelligence*, vol. 2, pp. 55–78. Morgan Kaufmann, San Francisco (1989)
3. Levy, R., Rosenschein, J.S.: A Game Theoretic Approach to Distributed Artificial Intelligence and The Pursuit Problem. In: *Proceedings of the 3rd European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, pp. 129–146 (1992)
4. Shannon, C.E.: *The Mathematical Theory of Communication*. University of Illinois Press (1949)
5. Tan, M.: Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In: *Proceedings of the 10th International Conference on Machine Learning*, pp. 330–337 (1993)
6. Watkins, C.J.H., Dayan, P.: Technical note: Q-learning. *Machine Learning* 8, 55–68 (1992)
7. Parunak, H.V.D., Brueckner, S.: Entropy and Self-Organization in Multi-Agent Systems”. In: *Proceedings of fifth International Conference on Autonomous Agents*, pp. 124–130 (2001)

Reo Connectors as Coordination Artifacts in 2APL Systems

Farhad Arbab^{1,2}, Lăcrămioara Aștefănoaei², Frank S. de Boer^{1,2},
Mehdi Dastani³, John-Jules Meyer³, and Nick Tinnermeier³

¹ Universiteit Leiden, The Netherlands

² CWI, Amsterdam, The Netherlands

³ Universiteit Utrecht, The Netherlands

Abstract. Communication is an important topic in multi-agent systems. Reo is a channel-based exogenous coordination model which can be used not only for the communication, but also for the coordination of individual agents. 2APL is an agent-oriented programming language where at the multi-agent level one can specify for individual agents which external environments they have access to. In this paper, we understand Reo networks as specific environments for 2APL agents and we focus on the integration of Reo networks into the 2APL platform. We show how Reo networks can be used to coordinate the behaviour of individual agents by means of an auction example.

1 Introduction

An important issue in agent software development is communication and coordination. In this paper, we focus on the integration of a channel-based communication mechanism into an agent platform. Channel-based communication enjoys many of the benefits of other communication models (messaging, shared spaces, events). It allows *efficient* implementations of point-to-point models, it supports *private* communication, it is *architecturally expressive* and it is *anonymous*. There are several kinds of channel-based communication models, MoCha [8], Nomadic Pict [10], Reo [1, 3] to name a few. The middleware MoCha, for example, has the advantage of being a real infrastructure for distributed implementations. However, we choose the Reo language since we can use channels not only for communication, but also for building complex connectors by composing channels. Such connectors impose specific coordination patterns on the execution of individual agents.

We approach the problem of communication (coordination) in multi-agent systems by fixing 2APL [5] as the language for implementing agents, Reo as the language for implementing coordination artifacts in the 2APL framework, and by focusing on their integration. The 2APL framework consists of agents and environments to which the agents have access. We understand nodes in a Reo network as a particular environment through which the agents can communicate. Furthermore by connecting channels in a particular way, we obtain specific

infrastructures on top of the nodes, which can be used as a coordination mechanism which, for example, restricts the execution of the agents. One important feature of the Reo language lies in the concept of “exogenous coordination”. In this way, we obtain a clear separation between execution (of agent programs) and control (of executions). Finally, we note that there exists a wreath of tools, “The Eclipse Coordination Tools”¹, which provide facilities for designing and verifying Reo networks.

The idea of using Reo as a coordination language for agents is not new, it appears first in [6]. However, what we offer is an executable 2APL platform where it is possible to integrate Reo connectors as the underlying communication infrastructure and as coordination artifacts in 2APL systems. We illustrate the use of the presented integration by means of an auction scenario. Finally, we show that the presented integration provides tools that enable the verification of interaction and communication properties.

In this paper, we briefly present the 2APL platform in Section 2 and the Reo coordination language in Section 3. Section 4 describes the mechanism underlying the integration of 2APL and Reo, with a basic application as a case of study. Section 4.2 gives an insight on the application of model-checking and Section 5 concludes the paper.

2 2APL

2APL (A Practical Agent Programming Language) is an agent-oriented programming language that provides two distinguished sets of programming constructs to implement both multi-agent as well as individual agent concepts. At the multi-agent level one can specify which agents should be created and which external environments they have access to. At the individual agent level, one can specify each (to be created) agent in terms of declarative concepts such as beliefs and goals, and imperative concepts such as events and plans. 2APL multi-agent systems run on the 2APL platform², a development tool that is designed to support the implementation and execution of multi-agent systems programmed in 2APL. In this section we briefly describe those parts of the syntax and intuitive semantics of 2APL that are relevant to this paper. For a complete overview of the syntax and formal semantics of the 2APL programming language we refer to [5].

The specification at the multi-agent level indicates which type of agents and how many of them constitute a multi-agent system. Moreover, it indicates which external environments are involved in the multi-agent system and which agent can interact with which external environment. The syntax for multi-agent level specification is as follows:

$$\begin{array}{l} \text{agentname}_1 : \text{filename}_1 N_1 @\text{env}_1^1, \dots, \text{env}_1^n \\ \dots \\ \text{agentname}_p : \text{filename}_p N_p @\text{env}_p^1, \dots, \text{env}_p^m \end{array}$$

¹ The Eclipse Coordination Tools are at <http://homepages.cwi.nl/~koehler/ect/>

² The 2APL Platform can be downloaded at <http://cs.uu.nl/2apl>

Here agentname_i is the name of the agent to be created, filename_i is the file name in which the agent is specified, N_i is the number of such agents to be created (if $N_i > 1$, then the agent names will be indexed by a number), and @env_i^j is the name of the environment that the agents can access and interact with. Each environment env_i^j is specified by a Java class of which one instance is created by the 2APL platform when loading the multi-agent specification. We explain later in this section how such a Java class implements an environment. Next, we will describe the relevant concepts by which a 2APL agent is specified.

A 2APL agent has beliefs and goals. The beliefs of an agent represent the information the agent has (about itself and its environments). The goals of an agent specify the situation the agent wants to realize. The agent's beliefs are stored in its belief base, which is implemented as a Prolog program. The agent's goals are stored in its goal base as conjunctions of ground atoms. A 2APL agent has only goals it does not believe to have achieved. Consider, for example, the belief and goal base of an auction agent with the goal of having a bike.

```
Beliefs:
    bid(100). step(30). maximalBid(400).
Goals:
    have(bike)
```

In this example, the agent believes its current bid to be EUR 100 (at the first round this is its initial bid), and will increase its bid each round with EUR 30 to a maximum of EUR 400. As soon as the agent believes to have bought the bike (i.e., `have(bike)` can be derived from the agent's beliefs) this goal is removed from the goal base.

To achieve its goals an agent needs to act. Actions that a 2APL agent can perform include actions to update its beliefs, actions to query its belief and goal base and external actions to interact with its environment.

The belief base of the agent can be updated by the execution of a belief update action. Such belief updates are specified in terms of pre- and post-conditions. Intuitively, an agent can execute a belief update action if the pre-condition of the action is derivable from its belief base. After the execution of the action, the beliefs of the agent are updated such that the post-condition of the action is derivable from the agent's belief base. The belief update `UpdateBid(R)` for updating the current bid `bid(X)` to `bid(R)`, for example, is specified as:

```
{bid(X)} UpdateBid(R) {not bid(X), bid(R)}
```

A test action can be used to test whether the agent has certain beliefs and/or goals. A test action is a conjunction of belief and goal tests. Belief test actions are of the form $B(\phi)$ in which ϕ is a (Prolog) query to the belief base. Similarly, goal test actions are of the form $G(\phi)$ in which ϕ is a query to the goal base. A goal test action $G(\phi)$ is successful if ϕ is derivable from the agent's goal base. A (belief or goal) test action can result in a substitution for variables that occur in ϕ . For example, given the above belief base, the belief test action $B(\text{bid}(X))$ is successful resulting in the substitution $X/100$. A test action can be used in a plan to (1) instantiate variables in the subsequent actions of the plan (if the test

can be entailed by the agent's beliefs and goals), or (2) to block the execution of the plan (in case the test cannot be entailed by the agent's beliefs and goals).

A 2APL agent performs external actions to act upon its environment. In 2APL environments are implemented as Java classes of type `Environment`. External actions that can be performed in this environment are then to be implemented as methods of this class having a predefined signature:

```
Term actionName(Term t1, ..., Term tn)
```

in which `Term` is the Java equivalent of 2APL terms such as constants (numbers and idents) or variables. External actions in the 2APL programming language are then of the form:

```
@env(actionName(t1, ..., tn), R)
```

with `actionName(t1, ..., tn)` corresponding to the signature of the Java method implementing the action, `R` is the return value that can be used to capture (a part of) the result of the action, and `env` being the unique identifier of the `Environment` object that implements the environment. The performance of an external action then boils down to invoking the method specifying the external action and binding the return value of this method to `R`.

A 2APL agent adopts plans to achieve its goals. These plans are the recipes that describe which actions the agent should perform to reach the desired situation. In particular, plans are built of basic actions and can be composed (amongst others) by a sequence operator (i.e., `;`) and a conditional choice operator. Conditional choice operators are of the form `if φ then π_1 else π_2` . The conditional part of these expressions (φ) is a conjunction of belief tests $B(\phi)$ and goal tests $G(\phi)$ that are evaluated on the agent's beliefs and goals. Such a condition thus expresses that the agent should perform plan π_1 in case φ can be entailed by the agent's beliefs and goals and otherwise plan π_2 . Note that the conditional part of such an expression may result a substitution that binds some variables in the π_1 part of the expression.

An agent possibly has more than one plan to reach its goals. Which plans are the best usually depends on the current situation. Planning goal rules are used to generate plans based on the agent's goals and beliefs. Such a rule is typically of the form `head <- guard | body`. The head of the rule is a goal expression indicating whether the agent has a certain goal. The guard of the rule is a belief expression indicating whether the agent has a certain belief, and the body of the rule is the plan that can be used to achieve the goal as stated by the head. A planning goal rule can be applied if the head and guard of the rule can be entailed by the agent's beliefs and goals, respectively. As an example, consider the following planning goal rule of our auction agent:

```
have(X) <- not finished | { ... }
```

indicating that the plan between the brackets can be used to achieve the goal of having product `X` in case the agent believes the auction is not finished.

3 The Reo Coordination Language

This section provides a short presentation of Reo (further details can be found in [1, 3]). Reo is a channel-based exogenous coordination language wherein complex coordinators, called *connectors*, are built out of simpler ones. Reo can be understood as a “glue language” for compositional construction of connectors which represent coordination artifacts in component-based systems. The emphasis in Reo is on connectors and their composition, not on the components, which are seen as “black-boxes”. The connectors impose a specific behaviour on the components, without the knowledge of the internal structure of the components.

The mechanism for constructing connectors is channel composition. *Channels* are primitive connectors, with two ends which can be either *source* or *sink*. At a source end data enters the channel by performing a corresponding *write* operation, while at a sink end data leaves the channel by performing a corresponding *read* operation. Reo imposes no restriction on the behaviour of the channels and thus it allows an open-ended set of channel types with user-defined semantics. Figure 1 depicts the graphical representation of three basic channel types: **ab** is a synchronous channel (**Sync**), **cd** is a one buffer cell asynchronous FIFO channel (**FIFO1**), and **ef** is a synchronous drain channel (**SyncDrain**). Synchronous and FIFO channels have both a source and a sink end each. In a **Sync** channel data is simultaneously accepted at the source end and dispensed at the sink end. In a **FIFO1** channel data is accepted at the source only if the buffer is empty and data is dispensed at the sink end only if the buffer is full. **SyncDrain** channels have two source ends and no sink. In a **SyncDrain** channel data is simultaneously accepted at the source ends and then destroyed.

Channels are composed via a join operation in a node which consists of a set of channel ends. Such a node is either source, sink, or mixed depending on whether all channel ends which coincide on the node are only source, only sink or a combination of source and sink. Source and sink nodes represent input and output ports where components connect to the network. A component can write data to a source node (input port) only if all source ends coincident on the node accept the data, in which case the data is written on each source end. Source nodes, thus, replicate data. A component can obtain data from a sink node (output port) only if at least one of the sink nodes coincident on the node offers data. In the case of more offers one is chosen nondeterministically. Sink nodes, thus, nondeterministically merge data. We take as an example the Reo diagram shown in Figure 2. This diagram represents a Reo connector which we use later in the paper. It implements a barrier synchronisation: by definition, the **SyncDrain** channel **ef** ensures that a data item passes from **ab** to **cd** only simultaneously with the passing of a data from **gh** to **ij** (and vice-versa).

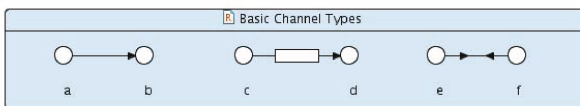


Fig. 1. Basic Channel Types in Reo

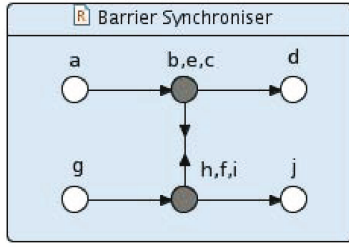


Fig. 2. A Barrier Synchroniser Connector

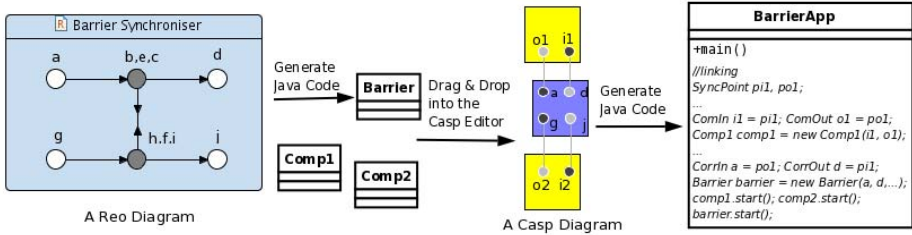


Fig. 3. Implementing a Component-Based System with a Barrier Synchroniser

3.1 Reo in Practice

In this section we briefly describe the process of building up a custom Java application which implements a component-based system coordinated by a Reo connector. The design mechanism relies on a bundle of plugins for the Eclipse development environment called “The Eclipse Coordination Tools”.

We take as an illustration a system with two components which simply alternate write and read operations. We assume that the behaviour of one component is to first write to the source node **a** and then read from the sink node **g**, and the same for the other one (a write to **g** is followed by a read from **j**). We further assume that the writings are controlled by a barrier synchroniser, and in this way, we have a simple mechanism of coordinating the components. Basically, the programmer starts by drawing the Reo connector from Figure 2 using the Reo editor. This diagram is automatically converted to the Java code which we denote as **Barrier** in Figure 3. Given that the components are implemented as Java threads (**Comp1** and **Comp2** in Figure 3), the programmer simply needs to drag and drop their corresponding code and the code for the barrier synchroniser into the Casp editor, which is meant to facilitate the programmer to wire the components to the coordinator. After the linking is completed, the system automatically generates code that implements the whole application (i.e., it generates a Java class where the constructors for **Comp1**, **Comp2**, **Barrier** are properly instantiated and the corresponding threads are started). Note that connectors can be exported as Reo libraries which can be later on reused. A growing collection of commonly useful connectors already exists.

4 Integrating Reo Connectors into the 2APL Platform

In this section we describe a mechanism for integrating Reo connectors into the 2APL platform. For this we consider a particular environment `reo`. The execution of any 2APL external action in the environment `reo` is a read from or a write to a given sink or source node, respectively. It is the task of the MAS programmer to setup the links between 2APL action names and Reo nodes. This should be done in a configuration class, which, in this section, we call `ReoCustom`. This class should be understood as an interface between the Reo network and 2APL agents. The MAS programmer should bear in mind that the association of an action name to a source node is to be interpreted as a write operation to the node. Similarly, the association of an action name to a sink node is to be interpreted as a read operation from the node. We take, for instance, the following setup. We assume that the MAS programmer creates a MAS file with the specification `bidder1 : bidder1.2apl @reo` and that the 2APL code for the agent `bidder1.2apl` contains the external action call `@reo(bid(100),_)`. This means that there exists a corresponding node in the Reo network. Let this node be a source node `p4`. Under these assumptions, if the MAS programmer wants to implement that `@reo(bid(100),_)` is a write on `p4`, then he or she needs to associate the action `bid` of `bidder1` to `p4`. This association is done in the `ReoCustom` configuration class by the following statement:

```
addSourceNode('bidder1','bid',p4)
```

where the parameters are the name of the agent, the name of the 2APL action and a source node. Similarly, the association of an action name with a sink node is done by calling `addSinkNode`. These functions are implemented in a specific environment `ReoEnvironment`. Besides providing functions which facilitate the MAS programmer to make the associations between action names and nodes, `ReoEnvironment` has a further use as well. Please note that the `@reo` external actions have a generic execution mechanism (either a read from or a write to a given node). It follows that it is desirable that the MAS programmer is spared the trouble of implementing them (as it is the case with external actions in general). `ReoEnvironment` is designed especially to make the implementation of `@reo` external actions transparent to the MAS programmer.

We conclude the section with a small remark. So far we have left the “wiring” up to the MAS programmer. However, we can imagine other options through which the interface is created automatically. For instance, we could think of scripting languages, where one could even design mechanisms which support parametrised MAS files as input. It should also be possible to use the Casp editor (see Section 3.1) as such an alternative. For this, MAS files should specify for each agent its interface to the Reo network. For example, the MAS file

```
bidder1 : bidder1.2apl @reo (bid p4) (readMax p3)
```

specifies that `bidder1` can perform the external actions `bid` and `readMax` in the environment `reo`. Furthermore, it specifies that the action `bid` is associated with the node `p4`, and `readMax` with `p3`. Such an approach has the advantage that a node could be associated with more than one action.

4.1 An Auction Scenario

In this section we propose an auction scenario illustrating the use of Reo based coordination artifacts in a 2APL system. We assume that we have a set of agents taking part in a sealed-bid auction. Each agent has its own maximal bid and its own strategy of increasing the bid. All participants submit their initial bid at the same time, then they wait for a response with the highest bid. If they want to continue they increase the highest bid with their chosen amount, otherwise they submit a default value 0. The auction ends when all minus one of the participants submit 0. The winner is the one with a non-zero bid. Typically, the planning goal rule of such a bidder is implemented in 2APL as follows:

```

have(X) <- highestBid(H) and maximalBid(Max) and step(S) and
    bid(C) and oldBid(O) and not finished | {
    if B(Max > H + S)
    then { @reo(bid(H + S), _); UpdateBid(H + S) }
    else { @reo(bid(0), _) };
    @reo(readMax(nil), NH); UpdateHighestBid(NH);
    if B(highestBid(O) and oldBid(Y) and bid(Y))
    then Bought(X) else if B(highestBid(O)) then Finish() }

```

where `updateBid(X)`, `updateHighestBid(X)`, `Bought(X)`, `Finish()` are the internal actions of the bidder agent, defined simply as belief updates, and `bid(X)`, `readMax(X)` are the only external actions that bidders can perform in the environment reo. Assume that `auction.mas` is the MAS file describing two bidders, and assume that the bidding agents are implemented in `bidder1.2apl` and `bidder2.2apl`:

```

bidder1 : bidder1.2apl @reo
bidder2 : bidder2.2apl @reo

```

We implement the mechanism of the auction as a Reo connector. Whenever a bidder submits a bid, a writing to the corresponding node occurs. We ensure that the bids happen simultaneously by using the barrier synchroniser described in Section 3, as it can be noticed in Figure 4.

Adding components to a multi agent system has the advantage of making our approach more powerful, generic and modular. We advocate the use of components whenever a standard task, with a clear meaning, needs to be implemented.

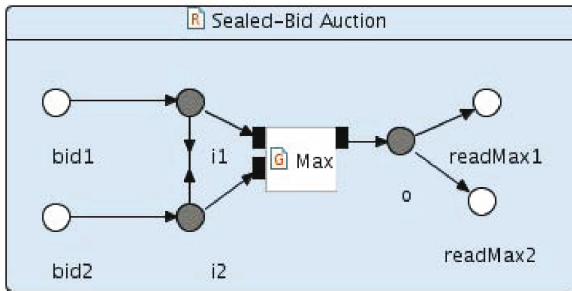


Fig. 4. A Reo Connector Implementing an Auction

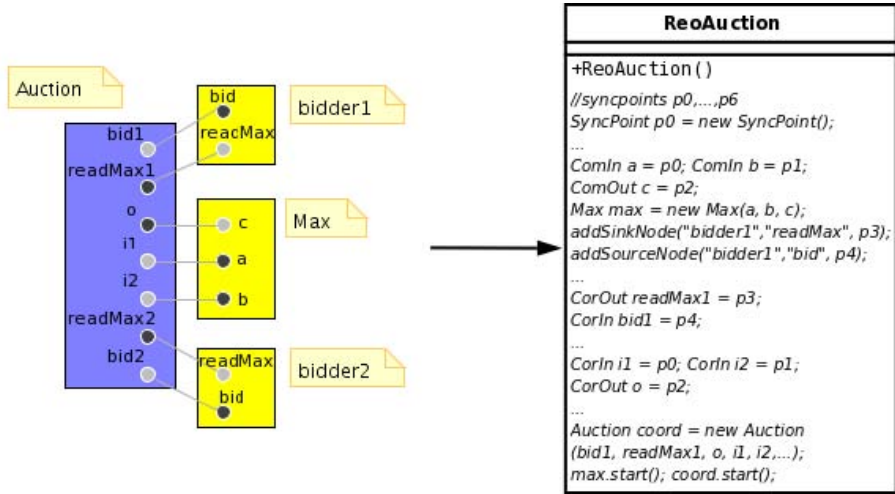


Fig. 5. ReoAuction: The Interface between the Bidders and the Connector Auction

This is why we choose to implement the auctioneer as a component, **Max**, which basically takes the data from its input nodes and forwards the maximum value. The value computed by **Max** is broadcast to **readMax1** and **readMax2**, which coincide with the sink nodes associated to the action **readMax** of the bidders, thus the bidders can read the value of the highest bid and continue the auction.

Given that we generated from the Reo diagram in Figure 4 a Java class **Auction** by the mechanism described in Section 3.1, we can now proceed with filling in the missing information in the interface we referred to as **ReoCustom** in Section 4. Since it is application dependent, we name it **ReoAuction**. This is the place where we setup the links between the nodes of the coordinator and the nodes of all other components, in our case, the bidding agents and **Max**. This is partially done by generating code from the Casp diagram (see Figure 5). We further need to setup the associations between the external actions and the nodes of the bidders. Take, as an illustration, the function call `addSinkNode("bidder1", "readMax", p3)`, where **p3** is a synchronisation point representing, on the one hand, coordinator's source node **readMax1**, and on the other hand, **bidder1**'s sink node **readMax**. This establishes that whenever **bidder1** performs a **readMax** action it reads the data written to **readMax1**.

We assume that the first bidder has an initial bid of EUR 150, and that he is willing to increase the highest bid with the amount of EUR 10, until it reaches an upper limit of EUR 300. Similarly, we assume that the second bidder has an initial offer of EUR 100, that the increasing step is of EUR 30, and that the maximal bid is EUR 400. We also assume that both bidders have the goal of buying a bike. The implementation of the bidders' planning goal rule suggests that these are naive, as we can easily foresee the winner. The auction stops when

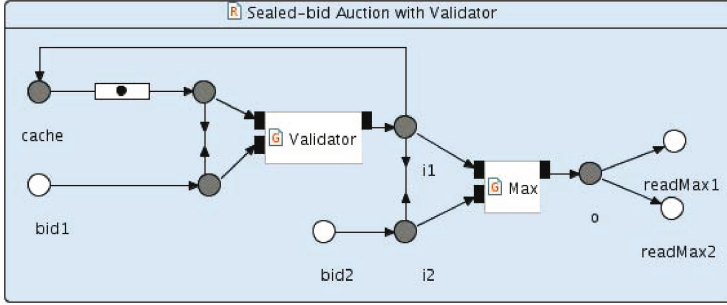


Fig. 6. The Auction Connector with a Validator for `bidder1`

the bidder with the smallest upper limit submits 0, in our case after `bidder1` bids EUR 290. This means that `bidder2` wins the bike for EUR 310. Running the application confirms our expectations.

For the sake of clarity, our scenario is on purpose simple. However, we could further make it more complex. For example, we could implement a component which validates the bids submitted by the agents: here we assume that the bidders always submit a higher bid than the previous one, however, this is a particular case. It is possible that the bidders have a different implementation, and that we might not even have access to the source code. It follows that it is desirable to impose a validation step in the Reo connector implementing the auction. Figure 6 shows the result of adding a validator component for `bidder1`.

The component `Validator` simply compares the bid submitted by the agents with the previous ones. In order to record previous bids and input them to the `Validator` we basically create a new node `cache` and a `Sync` channel connecting the input of `Max` to `cache`, such that each time `bidder1` submits a valid bid this value is fed to `cache`. The node `cache` is connected to the input of `Validator` through a full `FIFO1`, which initially, at the first round contains value 0. The `SyncDrain` channel ensures that the `Validator` component can fetch data only when the bidder submits a bid. Only if the bid is greater than the value read from the `cache` is the `cache` updated by inserting the bid, otherwise not. Note that in such a situation the flow of data through the connector is stopped.

4.2 Animating and Model-Checking Auctions

The Eclipse Coordination Tools are useful not only in designing Reo connectors but also in verifying them, as they contain an animation and a model-checker tool. The Reo animation is handy in the design phase. It helps the programmer to better understand the data flow in the Reo connector. The Reo model-checker [9] can be used in model-checking whether properties expressed in Branching Time Stream Logic (BTSL) are valid for the designed Reo coordination artifacts.

BTSL combines features of CTL [4], PDL [7] and time data stream logic (TDSL) [2]. We can therefore express properties like $\forall \langle bid_1 \wedge bid_2 \rangle true$. This means that for all executions, there exists a prefix which satisfies the constraint

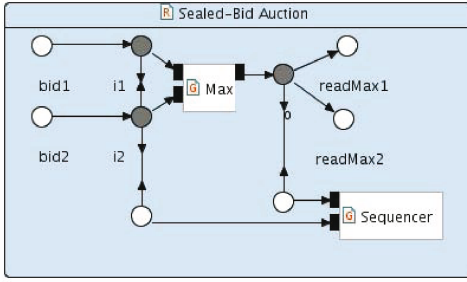


Fig. 7. The Auction Connector with a Sequencer

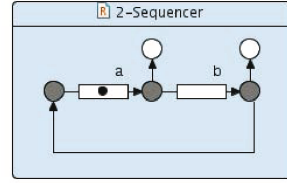


Fig. 8. A 2-Sequencer Connector

$bid_1 \wedge bid_2$ such that it reaches a state where $true$ holds. The constraint $bid_1 \wedge bid_2$ denotes that the operations on the nodes bid_1 and bid_2 happen simultaneously. The constraints in PDL-like formulas can also be defined on the data passing through nodes. A property like $\forall \langle d_{bid_1} = d_{i_1} \rangle true$ expresses that for all paths there exists a prefix where it holds that the data written at the node bid_1 (symbolically denoted by d_{bid_1}) is the same with the data d_{i_1} read at the node i_1 .

We can further extend our example by adding a sequencer connecting the sources and the sink of Max , as it can be noticed in Figure 7. A two place sequencer is described in Figure 8. It consists of two FIF01 and three Sync channels, with the leftmost FIF01 being initialised with a data item (the value is irrelevant). It ensures that the read operations can succeed only in the strict order from left to right. Such a connector is generic, one just needs to insert some more pairs of Sync and FIF01 channels in order to obtain a k -Sequencer.

Given the connector described in Figure 7 we can model-check that it can never be the case that the component Max outputs a higher bid before receiving the actual value of the highest bid: $\neg \exists \langle readMax_1; bid_1 \rangle true$. The regular expression $readMax_1; bid_1$ has precisely the meaning of “an operation on $readMax_1$ is followed by an operation on bid_1 ”. As one might expect, all the properties defined above hold for the connector from Figure 7.

Currently, no model checking tools exist for 2APL programs. However, using Reo encourages a compositional approach to the verification of systems, where (1) the externally observable behaviour of each 2APL agent is represented by a constraint automaton; (2) the system is verified as the product of the constraint automata of its agents and Reo connectors; and (3) the compliance of each individual agent with its constraint automaton model is verified separately.

5 Conclusions and Future Work

We have presented a mechanism for the integration of the existing Reo tools and 2APL platform. Such an integration enables the MAS programmer to incorporate Reo connectors as coordination artifacts into 2APL systems. Though

not presented in this paper, it is possible to have more than one Reo connector controlling the execution of the agents. Thus, scenarios like “agents participating in more than one auction at the same time” are indeed implementable. In such a way, the mas applications are more modular, distributed and multitasking.

We remark that the coordination patterns imposed by Reo connectors are not suitable for expressing organisational concepts like norms or sanctions. Thus, further work will focus on extending the 2APL platform such that conceptually different coordination artifacts are incorporated into 2APL systems. Another long term project concerns scalability issues. Currently, the number of the agents specified in a mas file is a priori fixed, and the same holds for the elements of a Reo network. However, there is current work on dynamic reconfiguration of Reo networks which could be integrated such that the network copes with agents entering and leaving a mas system.

References

- [1] Arbab, F.: Reo: a channel-based coordination model for component composition. *Mathematical Structures in Computer Science* 14(3), 329–366 (2004)
- [2] Arbab, F., Baier, C., de Boer, F., Rutten, J.: Models and temporal logics for timed component connectors. In: *SEFM 2004: Proceedings of the Software Engineering and Formal Methods, Second International Conference, Washington, DC, USA*, pp. 198–207. IEEE Computer Society Press, Los Alamitos (2004)
- [3] Baier, C., Sirjani, M., Arbab, F., Rutten, J.: Modeling component connectors in reo by constraint automata. *Sci. Comput. Program* 61(2), 75–113 (2006)
- [4] Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.* 8(2), 244–263 (1986)
- [5] Dastani, M.: 2APL: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems* 16(3), 214–248 (2008)
- [6] Dastani, M., Arbab, F., de Boer, F.: Coordination and composition in multi-agent systems. In: *AAMAS 2005: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 439–446. ACM Press, New York (2005)
- [7] Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.* 18(2), 194–211 (1979)
- [8] Guillen-Scholten, J., Arbab, F., de Boer, F., Bonsangue, M.: Mocha-pi, an exogenous coordination calculus based on mobile channels. In: *SAC 2005: Proceedings of the 2005 ACM symposium on Applied computing*. ACM Press, New York (2006)
- [9] Klüppelholz, S., Baier, C.: Symbolic model checking for channel-based component connectors. *Electron. Notes Theor. Comput. Sci.* 175(2), 19–37 (2007)
- [10] Wojciechowski, P.T., Sewell, P.: Nomadic pict: Language and infrastructure design for mobile agents. *IEEE Concurrency* 8(2), 42–52 (2000)

A Verification Framework for Normative Multi-Agent Systems

Lăcrămioara Aștefănoaei¹, Mehdi Dastani²,
John-Jules Meyer², and Frank S. de Boer¹

¹ CWI, Amsterdam, The Netherlands

² Universiteit Utrecht, The Netherlands

Abstract. This paper presents a programming language that facilitates the implementation of coordination artifacts which in turn can be used to regulate the behaviour of individual agents. The programming language provides constructs inspired by social and organisational concepts. The operational semantics of the language is prototyped in Maude, a rewrite logic software. Properties of the coordination artifacts are model-checked with the Maude LTL model-checker.

1 Introduction

One of the challenges in the design and development of multi-agent systems is to coordinate and control the behaviour of individual agents. Some approaches aim at achieving this objective by means of exogenous coordination artifacts, which are designed and built in terms of concepts such as action synchronisation and resource access relation [1,2]. Other approaches advocate the use of social and organisational concepts (e.g., norms, roles, groups, responsibility) and mechanisms (monitoring agents' actions and sanctioning mechanisms) to organise and control the behaviour of individual agents [3]. Yet other approaches aim at combining these by proposing organisation-based coordination artifacts, i.e., coordination artifacts that are designed and developed in terms of social and organisational concepts [4,5]. In such combined approaches, a multi-agent system is designed and developed in terms of an organisation artifact and the constituting individual agents. In order to ensure that the developed multi-agent systems achieve their overall design objectives and satisfy some global desirable properties, one has to verify the organisation artifact that constitutes the coordination and control part of the multi-agent system.

In this paper, we present a verification framework for normative multi-agent systems in which individual agents are coordinated and controlled by norm-based organisation artifacts. Such artifacts refer to norms as a way to signal when violations take place and sanctions as a way to respond (by means of punishments) in the case of violations. Basically, a norm-based artifact observes the actions performed by the individual agents, determines their effects in the environment (which is shared by all individual agents), determines the violations caused by performing the actions, and possibly, imposes sanctions.

We first present a programming language that is designed to facilitate the implementation of norm-based organisation artifacts. The operational semantics of the language makes it easy to prototype it in Maude [6], a rewriting logic software. Rewriting logic can be used as a computational framework for modular programming language design and formal analysis [7]. It has the benefit that there is *no gap between semantics and implementation*. This has a great advantage over the situations where one needs to implement an interpreter in order to execute the semantics of the designed language. It is also the case that rewriting logic implementations like Maude offer prototype parsers for free since they support user-definable syntax. Another benefit is *rapid prototyping* of programming language designs. This makes it easier to experiment with new language constructions since one needs only to define and not also to implement them. Furthermore, rewriting logic offers a suite of *generic tools* for formal analysis, for instance, the Maude theorem prover and LTL model-checker, which can be used to prove properties of the language definitions.

2 Programming Normative Multi-Agent Systems

In this section, we present a programming language that facilitates the implementation of normative multi-agent systems. Individual agents are assumed to be implemented in a programming language, not necessarily known to the multi-agent system programmer, who is assumed to have a reference to the (executable) programme of each agent. Most noticeably, it is not assumed that the agents are able to reason about the norms of the system since we do not make any assumptions about the internals of individual agents. Agents perform their actions in an external environment which is part of and controlled by the organisation. Actions are assigned pre- and post-conditions. If the pre-condition holds in the current state of the environment (the execution of an action is enabled), then the multi-agent system organisation determines the effect of the action by updating the state with the facts which represent the post-condition. We consider norms as being represented by counts-as rules [8], which ascribe "institutional facts" (e.g. "a violation has occurred") to "brute facts" (e.g. "the agent is in the train without a ticket"). In our framework, brute facts constitute the factual state of the multi-agent organisation, which is represented by the environment (initially set by the programmer), while institutional facts constitute the normative state of the multi-agent organisation. The institutional facts are used with the explicit aim of triggering system's reactions (e.g. sanctions). Sanction rules determine what brute facts will be brought about by the system as a consequence of normative facts. Typically, such brute facts are sanctions, such as fines.

2.1 Syntax

In order to represent brute and institutional facts in our normative multi-agent system programming language, we introduce two disjoint sets of first-order atoms `<b-atoms>` and `<i-atoms>` to denote these facts. Moreover, we use `<ident>` to

```

<N-MAS_Prog> = "Agents:" ( <agentName> <agentProg> [<nr>] )+
              "Facts:" <bruteFacts>
              "Effects:" <effects>
              "Counts-As rules:" <counts-as>
              "Sanction rules:" <sanctions> ;
<bruteFacts>  = <b-literals> ;
<effects>     = ( "{" <b-literals> "}" <actName> "{" <b-literals> "}" )+ ;
<counts-as>   = ( <b-literals> "=>" <i-literals> )+ ;
<sanctions>   = ( <i-literals> "=>" <b-literals> )+ ;
<agentName>   = <ident> ;
<agentProg>   = <ident> ;
<nr>          = <int> ;
<actName>     = <ident> ;
<b-literals>  = <b-literal> {", " <b-literal>} ;
<i-literals>  = <i-literal> {", " <i-literal>} ;
<b-literal>   = <b-atom> | "not" <b-atom> ;
<i-literal>   = <i-atom> | "not" <i-atom> ;

```

Fig. 1. The EBNF syntax of normative multi-agent programs

denote a string and $\langle \text{int} \rangle$ to denote an integer. Figure 1 presents the syntax of the language in EBNF notation. A normative multi-agent system programme $N\text{-MAS_Prog}$ starts with the specification of agents by means of names ($\langle \text{agentName} \rangle$), references to the (executable) agent programmes ($\langle \text{agentProg} \rangle$), and the number of agents to be created ($\langle \text{nr} \rangle$). Next, one specifies the initial state of the environment by means of a set of first order literals. Effects are specified by means of action names, pre- and post conditions. For simplicity both counts-as and sanction rules are specified by implications. The antecedents of count-as rules denote either brute or institutional facts, while the consequents denote only institutional facts. This allows rules to indicate that a certain brute or institutional fact counts as another institutional fact. For example, speeding is a violation of traffic law (institutional fact), but this violation together with not paying the fine in time (brute fact) is considered as another violation (institutional fact). The antecedents of sanction rules consist of literals denoting institutional facts while the consequents of sanction rules consist of literals denoting brute facts. Figure 2 presents an example of a normative multi-agent system programme that implements a small part of a train system. The programme creates from the specification file `passager_prog` one agent `psg`. The **Facts** determine that `psg` is not in the train and has no ticket. The **Effects** indicate that `psg` performing `enter` when not in the train, results in `psg` being in the train (with or without a ticket). The **Counts-As** rules state that being in the train without having a ticket is a specific violation (`viol_ticket(X)`). The rule functions as an *enforcement* mechanism and it is based on the idea of responding to a violation such that the system returns to an acceptable state. However, there are situations where stronger requirements need to be implemented, for example, where it is never the case that `psg` enters the train without a ticket. This is what we call *regimentation* and in order to implement it we consider a


```

Agents:
    psg passenger_prog 1
Facts:
    -in_train(psg), -ticket(psg)
Effects: at_platform(X) }
        { -ticket(X) } buy_ticket(X) { ticket(X) }
        { -in_train(X) } enter(X) { in_train(X) }
at_platform(X), in_train(X) } -in_train(X) }
Counts-As rules:
    in_train(X), -ticket(X) => viol_ticket(X)
Sanction rules:
    viol_ticket(X) => fined(X,25)

```

Fig. 2. An example of a Normative MAS file

specifically designated literal $\text{viol}_{\perp}(X)$. The operational semantics of the language ensures that the designated literal $\text{viol}_{\perp}(X)$ can never hold during any run of the system. Intuitively, rules with $\text{viol}_{\perp}(X)$ as consequence could be thought of as placing gates blocking an agent's action. Finally, the aim of **Sanction** rules is to determine that the violation of type $\text{viol_ticket}(X)$ causes the sanction $\text{fined}(X,25)$ (e.g., a 25 EUR fine).

2.2 Operational Semantics

The state of a normative multi-agent system consists of the state of the external environment, the normative state of the organisation, and the states of individual agents.

Definition 1 (Normative MAS Configuration). *Let P_b and P_n be two disjoint sets of first-order literals denoting atomic brute and normative facts (including viol_{\perp}), respectively. Let A_i be the configuration of individual agent i . The configuration of a normative multi-agent system is defined as $\langle \mathbf{A}, \sigma_b, \sigma_n \rangle$ where $\mathbf{A} = \{A_1, \dots, A_n\}$, σ_b is a consistent set of ground literals from P_b denoting the brute state of the multi-agent system, and σ_n is a consistent set of ground literals from P_n denoting the normative state of the multi-agent system.*

Before presenting the transition rules for specifying possible transitions between normative multi-agent system configurations, we need to define the ground closure of a set of literals (e.g., literals representing the environment) under a set of rules (e.g., counts-as or sanction rules¹) and the update of a set of ground literals (representing the environment) with another set of ground literals based on the specification of an action's effect. Let $l = (\Phi(\bar{x}) \Rightarrow \Psi(\bar{y}))$ be a rule, where Φ and Ψ are two sets of first-order literals in which sets of variables \bar{x} and \bar{y} occur. We assume that $\bar{y} \subseteq \bar{x}$ and that all variables are universally quantified

¹ Counts-as and sanctions are usually considered as being context dependent. Our framework can easily be extended by considering both rule types in a non-monotonic way capturing their context dependence.

in the widest scope. In the following, cond_l and cons_l are used to indicate the condition Φ and the consequent Ψ of the rule l , respectively. Given a set \mathbf{R} of rules and a set X of ground literals, we define the set of applicable rules in X as:

$$\text{App1}^{\mathbf{R}}(X) = \{ (\Phi(\bar{x}) \Rightarrow \Psi(\bar{y}))\theta \mid \Phi(\bar{x}) \Rightarrow \Psi(\bar{y}) \in \mathbf{R} \wedge X \models \Phi\theta \}$$

where θ is a ground substitution.

The ground closure of X under \mathbf{R} , denoted as $\text{C1}^{\mathbf{R}}(X)$, is inductively defined as follows:

$$\begin{aligned} \mathbf{B} : \text{C1}_0^{\mathbf{R}}(X) &= X \cup (\bigcup_{l \in \text{App1}^{\mathbf{R}}(X)} \text{cons}_l) \\ \mathbf{S} : \text{C1}_{n+1}^{\mathbf{R}}(X) &= \text{C1}_n^{\mathbf{R}}(X) \cup (\bigcup_{l \in \text{App1}^{\mathbf{R}}(\text{C1}_n^{\mathbf{R}}(X))} \text{cons}_l) \end{aligned}$$

We should emphasise that the counts-as rules obey some constraints. We consider only sets of counts-as rules such that 1) they are finite; 2) they are such that each condition has exactly one associated consequence (i.e., all the consequences of a given condition are packed in one single set cons); and 3) they are such that for counts-as rule k, l , if $\text{cond}_k \cup \text{cond}_l$ is inconsistent (i.e., contains p and $\neg p$), then $\text{cons}_k \cup \text{cons}_l$ is also inconsistent. That is to say, rules trigger inconsistent conclusions only in different states. Because of these properties (i.e., finiteness, consequence uniqueness and consistency) of counts-as rules \mathbf{R} one and only one finite number $m+1$ can always be found such that $\text{C1}_{m+1}^{\mathbf{R}}(X) = \text{C1}_m^{\mathbf{R}}(X)$ and $\text{C1}_{m-1}^{\mathbf{R}}(X) \neq \text{C1}_m^{\mathbf{R}}(X)$ and $\text{C1}_{m+1}^{\mathbf{R}}(X)$ is consistent. Let such $m+1$ define the ground closure X under \mathbf{R} , i.e., $\text{C1}^{\mathbf{R}}(X) = \text{C1}_{m+1}^{\mathbf{R}}(X)$.

In order to update the environment of a normative multi-agent system with the effects of an action performed by an agent, we use the specification of the action effect as implemented in the normative multi-agent system programme, unify this specification with the performed action to bind the variables used in the specification, and add/remove the resulting ground literals of the post-condition of the action specification to/from the environment. In the following, we assume a function *unify* that returns the most general unifier of two first-order expressions.

Definition 2 (Update Operation). *Let \bar{x}, \bar{y} , and \bar{z} be sets of variables whose intersections may not be empty, $\varphi(\bar{y}) \alpha(\bar{x}) \psi(\bar{z})$ be the specification of the effect of action α , and $\alpha(\bar{t})$ be the actual action performed by an agent, where \bar{t} consists of ground terms. Let σ be a ground set of literals, $\text{unify}(\alpha(\bar{x}), \alpha(\bar{t})) = \theta_1$, and $\sigma \models \varphi(\bar{t})\theta_1\theta_2$ for some ground substitution θ_2 . Then, the update operation is defined as follows:*

$$\begin{aligned} \text{update}(\sigma, \alpha(\bar{t})) &= \sigma \setminus \{ \Phi \mid \Phi \in \psi(\bar{z})\theta_1\theta_2 \wedge \text{NegLit}(\Phi) \} \\ &\cup \{ \Phi \mid \Phi \in \varphi(\bar{x})\theta_1\theta_2 \wedge \text{PosLit}(\Phi) \} \end{aligned}$$

In this definition, the variables occurring in the post-condition of the action specification are first bound and the resulted ground literals are then used to update the environment. Note that negative literals from the post-condition (i.e., $\text{NegLit}(\phi)$) are removed from the environment and positive literals (i.e., $\text{PosLit}(\phi)$) are added to it.

We do not make any assumptions about the internals of individual agents. Therefore, for the operational semantics of normative multi-agent system we assume $A_i \xrightarrow{\alpha(i,\bar{t})} A'_i$ as being the transition of configurations for individual agent i . Given this transition, we can define a new transition rule to derive transitions between normative multi-agent system configurations.

Definition 3 (Transition Rule). *Let $\langle \mathbf{A}, \sigma_b, \sigma_n \rangle$ be a configuration of a normative multi-agent system. Let \mathbf{R}_c be the set of counts-as rules, \mathbf{R}_s be the set of sanction rules, and α be an external action. The transition rule for the derivation of normative multi-agent system transitions is defined as follows:*

$$\frac{A_i \xrightarrow{\alpha(i,\bar{t})} A'_i \quad \sigma'_b = \text{update}(\sigma_b, \alpha(\bar{t})) \quad \sigma'_n = \text{Cl}^{\mathbf{R}_c}(\sigma'_b) \setminus \sigma'_b \quad \sigma'_n \not\models \text{viol}_\perp}{\langle \mathbf{A}, \sigma_b, \sigma_n \rangle \rightarrow \langle \mathbf{A}', \sigma'_b \cup S, \sigma'_n \rangle} \quad (\text{NMas-act})$$

where $A_i \in \mathbf{A}$, $\mathbf{A}' = (\mathbf{A} \setminus \{A_i\}) \cup \{A'_i\}$ and viol_\perp is the designated literal for regimentation.

The transition rule (*NMas-act*) captures the effects of performing an external action by an individual agent on both external environments and the normative state of the multi-agent system. First, the effect of $\alpha(\bar{t})$ on the environment σ_b is computed. Then, the updated environment is used to determine the new normative state of the system by applying all counts-as rules to the new state of the environment. Finally, possible sanctions are added to the environment state by applying sanction rules to the new normative state of the system. Note that the external action of an agent can be executed only if it does not result in a state containing viol_\perp . This captures exactly the regimentation of norms. Thus, once assumed that the initial normative state does not include viol_\perp , it is easy to see that the system will never be in a viol_\perp -state.

3 Prototyping Normative Multi-Agent Systems in Maude

In this section we describe a rewrite-based framework for modeling the programming language defined in Section 2. The main purpose and justification of our effort is verification. We want to be able to reason, on the one hand, about concrete normative multi-agent systems, and on the other hand, about the general semantics of the language. However, in this paper, we will deal only with properties of concrete normative multi-agent systems.

Rewriting logic is a logic of *becoming* and *change*, in the sense that it reasons about the evolution of concurrent systems. This follows from the fact that rewrite theories (the specifications of the rewriting logic) are defined as tuples $\langle \Sigma, E, R \rangle$, where Σ is a signature consisting of types and function symbols, E is a set of equations and R is a set of rewrite rules. The signature describes the states of the system, while the rewrite rules are executions of the transitions which model the dynamic of the system. Furthermore, the rules are intrinsically non-deterministic and this makes rewriting a good candidate for modeling concurrency.

We choose Maude as a rewriting logic language implementation since it is well-suited for prototyping operational semantics and since it comes with an LTL model-checker, on which we heavily rely in verification.

In what follows, we briefly present the basic syntactic constructions which are needed for understanding the remain of this section. Please refer to [6] for complete information. Maude programs are called *modules*. A module consists of *syntax declaration* and *statements*. The syntax declaration is called *signature* and it consists of declarations for *sorts*, *subsorts* and *operators*. The statements are either *equations*, which identify data, or *rewrite rules*, which describe transitions between states. The modules where the statements are given only by equations are called *functional modules*, and they define equational theories $\langle \Sigma, E \rangle$. The modules which contain also rules are called *system modules* and they define rewrite theories $\langle \Sigma, E, R \rangle$. Modules are declared using the keywords `fmod (mod)` `<ModuleName> is <DeclarationsAndStatements> endfm (endm)`. Modules can import other modules using the keywords `protecting`, `extending`, `including` followed by `<ModuleName>`. Module importation helps in building up modular applications from short modules, making it easy to debug, maintain or extend.

We now detail syntax declaration. The first thing to declare in a module is sorts (which give names for data types) and subsorts (which impose orderings on sorts). Take, for example, the declaration of a sort `Agent` as a subsort of `AgentSet`:

```
sorts Agent AgentSet . subsort Agent < AgentSet.
```

We can further declare operators (functions) defined on sorts (types) using the construction:

```
op <OpName> : <Sort-1> ... <Sort-k> -> <Sort>
[<OperatorAttributes>].
```

where `k` is the arity of the operator. Take, for instance, the following operator declarations:

```
ops a1 a2 : -> Agent . op *_ : AgentSet AgentSet -> AgentSet [comm
assoc].
```

where the operators `a1`, `a2` are constants (their arity is 0) of sort `Agent` and the associative and commutative operator `*` is a binary function in mixfix form (using underscores) with the meaning of a "union" operator. Variables are declared using the keyword `var`, for example `var A : Agent` represents the declaration of a variable `A` of sort `Agent`. Terms are either variables, or constants, or the result of the application of an operator to a list of argument terms which must coincide in size with the arity of the operator.

Equations are declared using one of the following constructions, depending on whether they are meant to be conditional:

```
eq [<Label>] : <Term-1> = <Term-2> . ceq [<Label>] : <Term-1> =
<Term-2> if <Cond-1> /\ ... /\ <Cond-k>.
```

where **Cond-i** is a condition which can be in turn either an ordinary equation $t = t'$, a matching equation $t := t'$ (which is true only if the two terms match), or a Boolean equation (which contain, e.g., the built-in (in)equality \neq , $=$, and/or logical combinators such as **not**, **and**, or). As an illustration the conditional equation labeled **init** defines the constant **agS** as a set with two elements **a1**, **a2** in the case the elements are distinct:

```
op agS : -> AgentSet . ceq [init] : agS = a1 * a2 if a1 /= a2.
```

Rewrite rules can also be conditional. Their declaration follows one of the patterns:

```
r1 [<Label>] : <Term-1> => <Term-2> . cr1 [<Label>] : <Term-1> =>
<Term-2> if <Cond-1> /\ ... /\ <Cond-k>.
```

where **Cond-i** can involve equations, memberships (which specify terms as having a given sort) and other rewrites. Take, for instance, the rule **step**:

```
cr1 [step] : agS => a' * a2 if a1 => a'.
```

which states that **agS** changes if **a1** is rewritten to **a'**, where we consider **a'** as a constant of sort **Agent**.

3.1 Executable Normative Multi-Agent Systems

We prototype the language for normative multi-agent systems in two modules: the first one, which we call **SYNTAX**, is a functional module where we define the syntax of the language, and the latter, which we call **SEMANTICS**, is a system module where we implement the semantics, namely the transition rule (*NMas-act*).

We recall that the state of a normative multi-agent system is constituted by the states of the agents together with the set of brute facts (representing the environment) and normative facts. The following lines, extracted from the module **SYNTAX**, represent the declaration of a normative multi-agent system and the types on which it depends:

```
sorts BruteFacts NormFacts NMasState . op <_,_,_> : AgentSet
BruteFacts NormFacts -> NMasState.
```

The brute (normative) facts are sets of ground literals. The effects are implemented by means of two projection functions, **pre** and **post** which return the enabling condition and the effect of a given action executed by a given agent:

```
op pre : Action Qid -> Query . op post : Action Qid -> LitSet.
```

Norms or sanctions are implemented similarly. Both have two parameters, an element of type **Query** representing the conditions, and an element of type **LitSet** representing the consequents. Take, for example, the declaration of **norm(s)**:

```
sorts Norm Norms . op norm : Query LitSet -> Norm . subsorts Norm <
Norms . op *_ : Norms Norms -> Norms [assoc comm].
```

The effect of a norm is to update the collection of normative facts whenever its condition matches either the set of brute facts or the set of normative facts:

```

op applyNorms : Norms Norms BruteFacts NormFacts NormFacts
  -> NormFacts.
ceq applyNorms(NS, norm(Q, E) * NS', BF, NF, OldNF) =
  applyNorms(NS, NS', BF, update(NF, E), NF)
if matches(Q, BF ; NF) /= noMatch.

```

where NS is an auxiliary variable which we need in order to compute the transitive closure of the normative set:

```

ceq applyNorms(NS, empty, BF, NF, OldNF) =
  applyNorms(NS, NS, BF, NF, NF) if NF /= OldNF.
eq applyNorms(NS, empty, BF, NF, NF) = NF [owise].

```

meaning that we apply the norms until no normative fact can be added anymore.

The application of norms entails the application of sanctions which, in a similar manner, update the brute facts when their conditions match the set of normative facts:

```

ceq applySanctions(SS, sanction(Q, E) * SS', NF, BF, OldBF) =
  applySanctions(SS, SS', NF, update(BF, E), BF)
if matches(Q, NF) /= noMatch.

```

Please note that we do not explain here the constructions `Action`, `Query`, `LitSet`, `update`, `matches`. This has already been done in [9] where we need such constructions for prototyping in Maude a BDI agent language which we call Belief Update programming Language (BUPL). For a better insight, we provide a basic web application illustrating the implementation at <http://homepages.cwi.nl/~astefano/agents/bupl-org.php>.

In a normative multi-agent system certain actions of the agents are monitored. Actions are defined by their pre- (enabling) and their post-conditions (effects). We recall the basic mechanism which takes place in the normative multi-agent system when a given monitored action is executed. First the set of brute facts is updated with the literals contained in the effect of the action. Then all possible norms are applied and this operation has as result an update of the set of normative facts. Finally all possible sanctions are applied and this results in another update of the brute facts. The configuration of the normative multi-agent system changes accordingly if and only if it is not the case that `violationReg`, the literal we use to ensure regimentation (corresponding to `viol⊥` in Section 2), appears in the brute facts. Consequently, the semantics of the transition rule (*NMas-act*) is implemented by the following rewrite rule:

```

crl [NMas-act] : < A * AS, BF, NF > =>
  < A' * AS, BF'; BF'', NF' >
if A => [Act] A'
/\ S := matches(pre(Act, Id), BF) /\ S /= noMatch
/\ BF' := update(BF, substitute(post(Act, Id), S))
/\ NF' := setminus(applyNorms(nS, nS, BF', NF, NF), BF')
/\ BF'' := setminus(applySanctions(sS, sS, BF', NF', BF'), NF')
/\ matches(violationReg(Id), NF') == noMatch.

```

where nS , sS are constants defined as the sets of instantiated norms, sanctions. Please note that we implement negation as *failure* and this implies that our update function preserves the consistency of the set of facts.

Given the above, we can proceed and describe how we can instantiate a concrete normative multi-agent system. We do this by creating a system module PSG-NMAS where we implement the constructions specified in Figure 2:

```

mod PSG-NMAS is
  including SEMANTICS .
  including BUPL-SEMANTICS .
  op psg : Qid BpMentalState -> Agent .
  eq pre(buy-ticket, X) = ~ has-ticket(X) .
  eq post(buy-ticket, X) = has-ticket(X) .
  eq pre(enter, X) = ~ in-train(X) .
  eq post(enter, X) = in-train(X) .
  op n : Qid -> Norm .
  eq [norm] : n(X) = norm(in-train(X) /\ ~ has-ticket(X),
    ticket-violation(X)) .
  op s : Qid -> Sanction .
  eq [sanction] : s(X) = sanction(ticket-violation(X),
    pay-fee-ticket(X)) .
  op nmas-state : Qid -> NMasState .
  eq [init] : nmas-state(X) = < psg(X), nil, nil > .
endm

```

The operator `psg` associates an identity to a BUPL agent. We stress that using BUPL agents is only a choice. Any other agent prototyped in Maude can be used instead. The actions being monitored are `buy-ticket`, `enter`, with obvious pre- and post-conditions. The equation `norm` defines a norm which introduces a ticket violation and the equation `sanction` introduces a punishment in the case of a ticket violation. We further consider that `psg` has a plan which consists of only one action, `enter`, meaning he enters the train without a ticket. This gives rise to special situations where model-checking turns out to be useful, as we will see in Section 3.2.

3.2 Model-Checking Normative Multi-Agent Systems

In order to model-check the system defined in the module PSG-NMAS we create a module PSG-NMAS-PREDS where we implement the predicates regimentation and enforcement as introduced in Section 2. Creating a new module is justified by the fact that state predicates are part of the *property specification* and should not be included in the *system specification*. In such a module we need to import two special modules LTL and SATISFACTION. Both are contained in the file `model-checker.maude` which must be loaded in the system before model-checking. We further need to make `NMasState` a subsort of the sort `State` which is declared in SATISFACTION. This enables us to define predicates on the states of a normative multi-agent system. A state predicate is an operator of sort `Prop`. The operator `op |=_ : State Formula -> Bool` is used to define the semantics of state predicates.

```

mod PSG-NMAS-PREDS is
  including PSG-NMAS .
  protecting SATISFACTION .
  extending LTL .
  subsort NMasState < State .
  op fact : Lit -> Prop .
  ceq < AS, BF, NF > |= fact(L) = true if in(L, BF) = true .
  ops enforcement regimentation : Qid -> Prop .
  eq [enf] : enforcement(X) =
    fact(in-train(X)) /\ not fact(has-ticket(X))
    -> <> fact(pay-fee-ticket(X)) .
  eq [reg] : regimentation(X) =
    [] (fact(in-train(X)) -> fact(has-ticket(X))).
endm

```

The state predicate `fact(L)` holds if and only if there exists a ground literal `L` in the set of brute facts of the normative multi-agent system. We need this predicate in order to define the properties enforcement and regimentation, which we are interested in model-checking. The equation `enf` defines the predicate `enforcement` such that it holds if and only if any agent `X` which is inside the train and has no ticket (`fact(in-train(X)) /\ not fact(has-ticket(X))` can be entailed from the brute facts) will eventually pay a fee. On the other hand, the equation `reg` defines the predicate `regimentation` such that it holds if and only if it is always the case that any agent in the train has a ticket.

If we model-check whether enforcement holds for an agent identified by `a1`:

```

Maude> red modelCheck(nmas-state('a1), enforcement('a1)) . reduce in
PSG-NMAS-PREDS :
  modelCheck(nmas-state('a1), enforcement('a1)).
result Bool : true

```

we obtain `true`, thus the normative structure enforces `a1` to pay a fee whenever it enters without a ticket. This is not the case for regimentation, the result of model-checking is a counter-example illustrating the situation where the agent enters the train without a ticket. This is because the implemented norm does not raise the special literal `violationReg(X)` (the already defined `viol⊥(X)`). However, if we replace in `PSG-NMAS` the equation `norm` by the following one:

```

eq [norm] : n(X) = norm(in-train(X) /\ ~ has-ticket(X),
violationReg(X)).

```

the application of the norm results in the update of the normative facts with `violationReg('a1)` and, in consequence, the rule `NMas-act` is not applicable and `nmas-state('a1)` is a deadlock state. It follows that the result of the model-checking is the boolean `true`, since `in-train('a1)` is not in the brute facts. We note that trivially regimentation would hold if the plan of `psg` consisted in buying a ticket before entering the train.

4 Conclusions and Future Work

We have presented a language for implementing normative multi-agent systems. In such a language one can implement organisational artifacts like norms and sanctions. These are meant to control/monitor the behaviour of individual agents. We have further prototyped the language in Maude, a rewriting logic software. This has the advantage of making it possible to model-check whether properties like enforcement or regimentation hold in a given normative multi-agent system.

So far, we have applied model-checking to given instances of normative multi-agent systems. However, it is also in our concern to define more generic properties which characterise normative multi-agent systems at a more abstract (higher) level. Further extensions of the language will be designed in the same idea which we have promoted in this paper, namely counter-pointed by verification in the Maude framework. Another direction for future work focuses on the integration of organisational artifacts in the existing 2APL platform [10]. 2APL is an agent-oriented programming language that provides two distinguished sets of programming constructs to implement both multi-agent as well as individual agent concepts. It is desirable to enrich it by incorporating such normative structures as the ones introduced in this paper.

References

1. Arbab, F.: Reo: a channel-based coordination model for component composition. *Mathematical Structures in Computer Science* 14(3), 329–366 (2004)
2. Ricci, A., Viroli, M., Omicini, A.: Give agents their artifacts: the a&a approach for engineering working environments in mas. In: Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O. (eds.) *AAMAS, IFAAMAS*, p. 150 (2007)
3. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multi-agent systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) *AOSE 2003. LNCS*, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
4. Esteva, M., Rosell, B., Rodríguez-Aguilar, J.A., Arcos, J.L.: Ameli: An agent-based middleware for electronic institutions. In: *AAMAS*, pp. 236–243. IEEE Computer Society, Los Alamitos (2004)
5. Hübner, J.F., Sichman, J.S., Boissier, O.: Moise+: towards a structural, functional, and deontic model for mas organization. In: *AAMAS*, pp. 501–502. ACM, New York (2002)
6. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: *All About Maude - A High-Performance Logical Framework. LNCS*, vol. 4350. Springer, Heidelberg (2007)
7. Serbanuta, T.F., Rosu, G., Meseguer, J.: A rewriting logic approach to operational semantics (extended abstract). *Electr. Notes Theor. Comput. Sci.* 192(1), 125–141 (2007)
8. Searle, J.R.: *The Construction of Social Reality*. The Penguin Press, London (1995)
9. Astefanoaei, L., de Boer, F.S.: Model-checking agent refinement. In: Padgham, L., Parkes, D.C., Müller, J., Parsons, S. (eds.) *AAMAS (2), IFAAMAS*, pp. 705–712 (2008)
10. Dastani, M.: 2apl: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems* 16(3), 214–248 (2008)

Social Viewpoints for Arguing about Coalitions

Guido Boella¹, Leendert van der Torre², and Serena Villata¹

¹ Dipartimento di Informatica, University of Turin, Italy
{boella,villata}@di.unito.it

² Computer Science and Communication, University of Luxembourg
leendert@vandertorre.com

Abstract. Frameworks for arguing about coalitions are based on non-monotonic logic and are therefore formal and abstract, whereas social theories about agent coalitions typically are based on conceptual modeling languages and therefore semi-formal and detailed. In this paper we bridge the gap between these two research areas such that social viewpoints can be used to argue about coalitions. We formally define three social viewpoints with abstraction and refinement relations among them, and we adapt an existing coalition argumentation theory to reason about the coalitions defined in the most abstract social viewpoint.

1 Introduction

Dung's argumentation theory [13] may be seen as a formal framework for nonmonotonic logic and logic programming, and has been applied to many domains in which non-monotonic reasoning plays a role, such as decision making or coalition formation [3]. Amgoud [1] proposes to use Dung's argumentation theory and associated dialogue theories as a formal framework for coalition formation, and she illustrates this idea by formalizing a task based theory of coalition formation as an instance of Dung's argumentation theory.

In this paper we develop social viewpoints for arguing about coalitions. Social viewpoints become more popular in multiagent systems, since the representation of multiagent systems as, for example, social networks, dependence networks, organizations, or normative systems, focuses on the interactions among the agents and facilitates the development of interaction mechanisms, agreement technologies or electronic institutions. Castelfranchi [12] offers a general and influential framework for many social-cognitive concepts and their relations, but due to the large number of concepts and their detailed descriptions, this framework is only semi-formal. Consequently, it can be used for conceptual modeling and conceptual analysis, but not for formal analysis or as the basis of formal ontologies or argumentation frameworks. In general, the problem with applying most existing social viewpoints is that they are often only semi-formal and relatively detailed, whereas the argumentation models of Dung and Amgoud are formal and abstract.

We therefore take our approach in [5] as a starting point, which not only defines social viewpoints on MAS, but also relates views of these viewpoints to each other using abstraction and refinement relations. For example, a detailed BDI model can be abstracted to a dependence network as used in early requirements analysis in Tropos [7].

In related work we show how to use these formal representations to define criteria for coalition formation [6], or measures for multiagent systems inspired by social network analysis, such as the social importance of an agent in a system [4]. However, the social viewpoints and the abstraction and refinement relations have been sketched only semi-formally in a two page paper, and moreover we consider only absolute goals and static dependence networks. In many agent programming languages, e.g. [14], and agent architectures, e.g. [9], goals can be derived from the agent's desires and beliefs, and in Castelfranchi's conceptual model, dependence relations can change over time.

In this paper we therefore address the following problems to increase the application of our social viewpoints on multiagent systems:

1. How to develop social viewpoints which can be used in arguing about coalitions?
How to define a dynamic dependence network for agents with conditional goals?
How to define coalitions for dynamic dependence networks? How to pass from the dynamic dependence networks view to the coalition view?
2. How to argue about coalitions using these social viewpoints? How to reason about attack relations among coalitions?

While the agent view represents the agents of the systems, their goals and the actions that they can perform to achieve these goals, the power view introduces conditional goals, such as those goals that can be added by an agent to another one. We define the dynamic dependence view as an abstraction of the power view, defined as a set of dependencies that can be added thanks to existence of conditional goals. Abstracting from the dynamic dependence view, we define the coalition view representing a coalition as a set of dynamic dependencies where each agent either creates a dependency or fulfills a goal.

In Amgoud's formalization, an argument is a set of agents together with a task, and an argument attacks another one if the two coalitions share an agent, or when they contain the same task. It is therefore based on strong assumptions, for example that an agent cannot be part of two coalitions at the same time. Since the attack relation is symmetric, also preferences are introduced to resolve conflicts. We need to use different viewpoints to describe coalitions since dynamic dependencies have two different roles. From the internal point of view, an agent inside a coalition can be described by viewpoints and this means that we can describe the coalition describing the agents and their goals and capabilities to achieve the goals (agent view) or we can describe the agents inside the coalition as a set of agents and the power relations that link these agents to each other (power view) or, finally, we can describe a coalition as a set of dynamic dependencies where an agent is made dependent on another one for a particular goal by means of the addition of a new dependence by a third agent. From the external point of view, instead, the addition of a new dependence can be represented as an attack relation from a coalition to another one at coalitional view level. In this paper we describe and reason about these attacks using argumentation theory.

The layout of this paper is as follows. In Section 2 we introduce the social viewpoints, and relate them to each other by abstraction and refinement relations. In Section 3 we introduce a modification of Amgoud's argumentation theory to use the social viewpoints.

2 Social Viewpoints

In classical planners, goals are unconditional. Therefore, many models of goal based reasoners including our earlier model [5] define the goals of a set of agents A by a function $goals : A \rightarrow 2^G$, where G is the complete set of goals. However, in many agent programming languages and architectures, goals are conditional and can be generated. We therefore extend the agent view with conditional goals.

Definition 1 (Agent view). *The Agent view is represented by the tuple $\langle A, G, X, goals, skills, R \rangle$, where:*

- A, G, X are disjoint sets of agents, goals, and decision variables,
- $goals : A \times 2^X \rightarrow 2^G$ is a function associating with an agent its conditional goals,
- $skills : A \rightarrow 2^X$ is a function associating with agents their possible decisions, and
- $R : 2^X \rightarrow 2^G$ is a function associating with decisions the goals they achieve.

Example 1

- $A = \{a, b, c, d\}$, $G = \{g_1, g_2, g_3, g_4\}$, $X = \{x_1, x_2, x_3, x_4, x_5\}$.
- $goals(a, \{\}) = \{g_4\}$, $goals(b, \{x_5\}) = \{g_3\}$, $goals(c, \{\}) = \{g_1\}$,
 $goals(d, \{\}) = \{g_2\}$.
- $skills(a) = \{x_5\}$, $skills(b) = \{x_1\}$, $skills(c) = \{x_2\}$, $skills(d) = \{x_3, x_4\}$.
- $R(\{x_1\}) = \{g_1\}$, $R(\{x_2\}) = \{g_2\}$, $R(\{x_3\}) = \{g_3\}$, $R(\{x_4\}) = \{g_4\}$.

The power to trigger a goal is distinguished from the power to fulfill a goal.

Definition 2 (Power view). *The Power view is represented by the tuple $\langle A, G, X, goals, power-goals, power \rangle$, where A, G and X are sets of agents, goals, and decision variables (as before), $goals : A \times 2^X \rightarrow 2^G$ is a function (as before), and:*

- $power-goals : 2^A \rightarrow 2^{(A \times G)}$ is a function associating with each set of agents the goals they can create for agents, and
- $power : 2^A \rightarrow 2^G$ is a function associating with agents the goals they can achieve.

The power view can be defined as an abstraction of the agent view, in other words, the agent view is a refinement of the power view. A set of agents B has the power to see to it that agent a has the goal g , written as $(a, g) \in power-goals(B)$, if and only if there is a set of decisions of B such that g becomes a goal of a . A set of agents B has the power to see to goal g if and only if there is a set of decisions of B such that g is a consequence of it.

Definition 3. $\langle A, G, goals, power-goals, power \rangle$ is an abstraction from $\langle A, G, X, goals, skills, R \rangle$ if and only if:

- $(a, g) \in power-goals(B)$ if and only if $\exists Y \subseteq skills(B)$ with $skills(B) = \cup\{skills(b) \mid b \in B\}$ such that $g \in goals(a, Y)$, and
- $g \in power(B)$ if and only if $\exists Y \subseteq skills(B)$ such that $g \in R(Y)$.

Example 2 (Continued). Agent a has no power to fulfill goals, but he can create a goal of agent d .

- $power\text{-}goals(\{a\}) = (\{d\}, \{g_3\})$
- $power(\{b\}) = \{g_1, g_3\}, power(\{c\}) = \{g_2\}, power(\{d\}) = \{g_4\}$.

Due to the power to create goals, dependence relations are no longer static, but they can be created by agents. We therefore have to extend the dependence networks developed by Conte and Sichman [23] and used in multiagent systems methodologies like Tropos [7]. Dynamic dependence networks can be defined as follows [11].

Definition 4 (Dynamic dependence view). A dynamic dependence network is a tuple $\langle A, G, \text{dyndep} \rangle$ where A and G are disjoint sets of agents and goals (as before), and:

- $\text{dyndep} : A \times 2^A \times 2^A \rightarrow 2^{2^G}$ is a function that relates with each triple of sets of agents all the sets of goals on which the first depends on the second, if the third creates the dependency.

We write $dep(a, B, G)$ for $\text{dyndep}(a, B, \emptyset) = G$.

Abstracting power view to a dynamic dependence network can be done as follows. Note that in this abstraction, the creation of a dynamic dependency is based only on the power to create goals. In other models, creating a dependency can also be due to creation of new skills of agents.

Definition 5. $\langle A, G, \text{dyndep} \rangle$ is an abstraction of $\langle A, G, \text{power-goals}, \text{power} \rangle$, if we have $H \in \text{dyndep}(a, B, C)$ if and only if

1. $\forall g \in H : (a, g) \in \text{power-goals}(C)$, and
2. $H \subseteq \text{power}(B)$

Example 3 (Continued). Agent b depends on agent d for goal g_3 , if agent a creates this dependency: $dep(a, d, g_4), dep(d, c, g_2), dep(c, b, g_1), \text{dyndep}(b, \{d\}, \{a\}) = \{\{g_3\}\}$.

Combining these two abstractions, abstracting agent view to a dynamic dependence network can be done as follows.

Proposition 1. $\langle A, G, \text{dyndep} \rangle$ is an abstraction of $\langle A, G, X, \text{goals}, \text{skills}, R \rangle$, if we have $H \in \text{dyndep}(a, B, C)$ if and only if

1. $\exists Y \subseteq \text{skills}(C)$ such that $H \subseteq \text{goals}(a, Y)$, and
2. $\exists Y \subseteq \text{skills}(B)$ such that $H \subseteq R(Y)$

Finally, we define reciprocity based coalitions for dynamic dependence networks. We represent the coalition not only by a set of agents, as in game theory, but as a set of agents together with a partial dynamic dependence relation. Intuitively, the dynamic dependence relation represents the “contract” of the coalition: if $H \in \text{dyndep}(a, B, D)$, then the set of agents D is committed to create the dependency, and the set of agents B is committed to see to the goals H of agent a . The rationality constraints on such reciprocity based coalitions are that each agent contributes something, and receives something back.

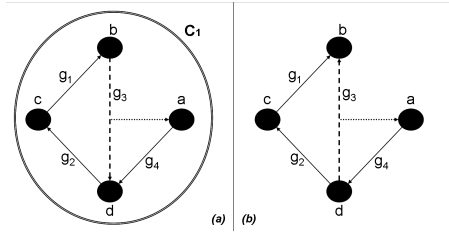


Fig. 1. (a) - The coalition C_1 of Example 4, (b) - Example of a set of agents that is not a coalition

Definition 6 (Reciprocity based Coalition). Given a dynamic dependence network $\langle A, G, \text{dyndep} \rangle$, a reciprocity based coalition is represented by coalition $C \subseteq A$ together with dynamic dependencies $\text{dyndep}' \subseteq \text{dyndep}$, such that

- if $\exists b, B, D, H$ with $H \in \text{dyndep}'(a, B, D)$ then $a \in C$, $B \subseteq C$ and $D \subseteq C$ (the domain of dyndep' contains only agents in coalition C), and
- for each agent $a \in C$ we have $\exists b, B, D, H$ with $H \in \text{dyndep}'(b, B, D)$ such that $a \in B \cup D$ (agent a contributes something, either creating a dependency or fulfilling a goal), and
- for each agent $a \in C$ $\exists B, D, H$ with $H \in \text{dyndep}(a, B, D)$ (agent a receives something from the coalition).

The following example illustrates that dependencies will be created by agents only if the new dependencies work out in their advantage.

Example 4 (Continued). Each agent of $C_1 = \{a, b, c, d\}$ creates a dependency or fulfills a goal. In Figure 1, conditional goals are represented with dashed arrows while the creation of new dependencies is represented with dotted ones. The arrows go from the agent having the goal put as label of the arrow to the agent that has the power to fulfill this goal. Figure 1 - (a) represents a set of agents composing a coalition in accordance with Definition 6 while Figure 1 - (b) represents the same set of agents not forming a coalition. The difference among the two figures is in the direction of the arrow joining agents b and d .

The basic attack relations between coalitions are due to the fact that they are based on the same goals. This is analogous to the conflicts between coalitions in Amgoud's coalition theory where two coalitions are based on the same tasks.

Definition 7. Coalition $\langle C_1, \text{dyndep}_1 \rangle$ attacks coalition $\langle C_2, \text{dyndep}_2 \rangle$ if and only if there exists $a_1, a_2, B_1, B_2, D_1, D_2, G_1, G_2$, such that $G_1 \in \text{dyndep}_1(a_1, B_1, D_1)$, $G_2 \in \text{dyndep}_2(a_2, B_2, D_2)$ and $G_1 \cap G_2 \neq \emptyset$.

The simplest kind of attack relation is to remove or add one of the dependencies of the attacker.

Definition 8. Coalition $\langle C, \text{dyndep} \rangle$ attacks the attack from coalition $\langle C_1, \text{dyndep}_1 \rangle$ on coalition $\langle C_2, \text{dyndep}_2 \rangle$ if and only if there exists a set of agents $D \subseteq \{a \mid \exists E, H, C(a, E, H)\}$ such that $\exists a, B, G' C_1(a, B, G')$ and $G \in \text{dyndep}(a, B, D)$.

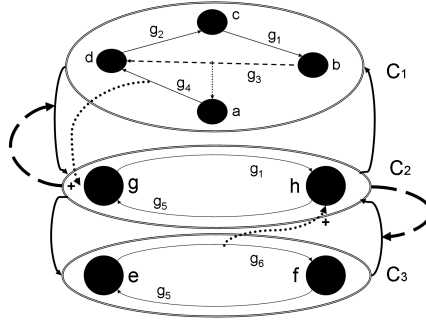


Fig. 2. Dynamic dependence view and coalition view

Example 5. Assume we have eight agents, a, b, c, d, e, f, g, h and the dependencies of Example 3: $dep(a, \{d\}, \{\{g_4\}\})$, $dep(d, \{c\}, \{\{g_2\}\})$, $dep(c, \{b\}, \{\{g_1\}\})$, $dyndep(b, \{d\}, \{a\}, \{\{g_3\}\})$,

plus the following ones:

$dep(e, \{f\}, \{\{g_6\}\})$, $dep(f, \{e\}, \{\{g_5\}\})$, $dep(g, \{h\}, \{\{g_1\}\})$, $dep(h, \{g\}, \{\{g_5\}\})$,
 $dep(c, \{h\}, \{\{g_1\}\})$, $dep(g, \{b\}, \{\{g_1\}\})$, $dep(h, \{e\}, \{\{g_5\}\})$, $dep(f, \{g\}, \{\{g_5\}\})$.

The possible coalitions are C_1 , C_2 and C_3 where:

$$C_1 = \{dep(a, \{d\}, \{\{g_4\}\}), dep(d, \{c\}, \{\{g_2\}\}), dep(c, \{b\}, \{\{g_1\}\}), dyndep(b, \{d\}, \{a\}, \{\{g_3\}\})\},$$

$$C_2 = \{dep(e, \{f\}, \{\{g_6\}\}), dep(f, \{e\}, \{\{g_5\}\})\},$$

$$C_3 = \{dep(g, \{h\}, \{\{g_1\}\}), dep(h, \{g\}, \{\{g_5\}\})\}.$$

Note that some of the dependencies remain outside all coalitions (e.g., $dep(c, \{h\}, \{\{g_1\}\})$, $dep(g, \{b\}, \{\{g_1\}\})$, $dep(h, \{e\}, \{\{g_5\}\})$, $dep(f, \{g\}, \{\{g_5\}\})$, not reported in Figure 2). Thus, $C_1 \# C_2$, $C_2 \# C_1$, $C_2 \# C_3$ and $C_3 \# C_2$ due to the fact that they share goals g_1 and g_5 respectively. Note that these attacks are reciprocal.

The coalitions attack each other since agents b and h on which respectively c and g depend for g_1 would not make their part hoping that the other one will do that, so to have a free ride and get respectively g_3 achieved by d and g_5 by g .

We depict this situation in Figure 2: normal arrows connecting the agents represent the dependencies among these agents (they can be labeled with the goal on which the dependence is based), coalitions are represented by the ovals containing the agents of the coalition, bold arrows indicate the attack relations among the coalitions (dashed bold arrows are explained in the subsequent example).

3 Arguing about Coalitions

Argumentation is a reasoning model based on constructing arguments, identifying potential conflicts between arguments and determining acceptable arguments. Amgoud [1] proposes to use it to construct arguments to form coalitions, identifying potential conflicts among coalitions, and determine the acceptable coalitions. Dung's framework [13]

is based on a binary attack relation among arguments. In Dung's framework, an argument is an abstract entity whose role is determined only by its relation to other arguments. Its structure and its origin are not known. In this section, following Amgoud, we assume that each argument proposes to form a coalition, but we do not specify the structure of such coalitions yet. We represent the attacks among arguments by $\#$.

Definition 9 (Argumentation framework). *An argumentation framework is a pair $\langle \mathcal{A}, \# \rangle$, where \mathcal{A} is a set (of arguments to form coalitions), and $\# \subseteq \mathcal{A} \times \mathcal{A}$ is a binary relation over \mathcal{A} representing a notion of attack between arguments.*

The various semantics of an argumentation framework are all based on the notion of defense. A set of arguments \mathcal{S} defends an argument a when for each attacker b of a , there is an argument in \mathcal{S} that attacks b . A set of acceptable arguments is called an *extension*.

Definition 10 (Acceptable arguments)

- $\mathcal{S} \subseteq \mathcal{A}$ is attack free if and only if there are no arguments $a_1, a_2 \in \mathcal{S}$ such that a_1 attacks a_2 .
- \mathcal{S} defends a if and only if for all $a_1 \in \mathcal{A}$ such that a_1 attacks a , there is an alternative $a_2 \in \mathcal{S}$ such that a_2 attacks a_1 .
- \mathcal{S} is a preferred extension if and only if \mathcal{S} is maximal with respect to set inclusion among the subsets of \mathcal{A} that are attack free and that defend all their elements.
- \mathcal{S} is a basic extension if and only if it is a least fix point of the function $F(\mathcal{S}) = \{a \mid a \text{ is defended by } \mathcal{S}\}$.

The following example illustrates argumentation theory.

Example 6. Let $AF = \langle \mathcal{A}, \# \rangle$ be an argumentation framework, where the set (of arguments or coalitions) is $\mathcal{A} = \{C_1, C_2, C_3\}$, and $\{C_1\#C_2, C_2\#C_3\}$ is the binary relation over \mathcal{A} representing a notion of *attack* between arguments. Due to the so-called reinstatement principle of argumentation theory, the acceptable arguments are C_1 and C_3 , for any kind of semantics. C_1 is accepted because it is not attacked by any other argument, and C_3 is accepted because its only attacker C_2 is attacked by an accepted argument.

Amgoud [1] proposes to use preference-based argumentation theory for coalition formation, in which the attack relation is replaced by a binary relation \mathcal{R} , which she calls a defeat relation, together with a (partial) preordering on the coalitions. Each preference-based argumentation framework represents an argumentation framework, and the acceptable arguments of a preference-based argumentation framework are simply the acceptable arguments of the represented argumentation framework.

Definition 11 (Preference-based argumentation framework). *A preference-based argumentation framework is a tuple $\langle \mathcal{A}, \mathcal{R}, \succeq \rangle$ where \mathcal{A} is a set of arguments to form coalitions, \mathcal{R} is a binary defeat relation defined on $\mathcal{A} \times \mathcal{A}$ and \succeq is a (total or partial) pre-order (preference relation) defined on $\mathcal{A} \times \mathcal{A}$. A preference-based argumentation framework $\langle \mathcal{A}, \mathcal{R}, \succ \rangle$ represents $\langle \mathcal{A}, \# \rangle$ if and only if $\forall a, b \in \mathcal{A}$, we have $a\#b$ if and only if $a\mathcal{R}b$ and it is not the case that $b \succ a$ (i.e., $b \succeq a$ without $a \succeq b$). The extensions of $\langle \mathcal{A}, \mathcal{R}, \succ \rangle$ are the extensions of the represented argumentation framework.*

The following example illustrates the preference based argumentation theory.

Example 7 (Continued). Let $PAF = \langle \mathcal{A}, \mathcal{R}, \succeq \rangle$ be a preference-based argumentation framework, where $\mathcal{A} = \{C_1, C_2, C_3\}$ is a set of arguments to form coalitions,

$$\{C_1 \mathcal{R} C_2, C_2 \mathcal{R} C_1, C_2 \mathcal{R} C_3, C_3 \mathcal{R} C_2\}$$

a binary defeat relation defined on $\mathcal{A} \times \mathcal{A}$ and $\{C_1 \succ C_2, C_2 \succ C_3\}$ a total order (preference relation) defined on $\mathcal{A} \times \mathcal{A}$. PAF represents AF , so the acceptable arguments are again C_1 and C_3 , for any kind of semantics.

In general, preference-based argumentation frameworks are a useful and intuitive representation for argumentation frameworks, but for the application of coalition formation it is less clear where the preferences among coalitions come from. Moreover, when the defeat relation is symmetric, as in Amgoud's task based coalition theory, then it leads to a lack of expressive power, because some attack cycles can no longer be represented (see [17] for details).

Modgil [18] relates preferences to second-order attacks. Suppose that arguments a and b attack each other, and that argument a is preferred to argument b . Modgil observes that we can then say that the preference attacks the attack relation from b to a . The advantage of this perspective is that Modgil introduces also arguments which attack attack relations, which he uses to represent non-monotonic logics in which the priorities among the rules are represented in the formalism itself, rather than being given a priori (such as Brewka's theory [8], or Prakken and Sartor's theory [20]). Whereas Modgil presents his theory as an extension of Dung, such that he has to define new semantics for it, in this paper we show how to define second order attacks as an instance of Dung's theory. Each second order argumentation framework represents an argumentation framework, and the acceptable arguments of the second order argumentation framework are simply the acceptable arguments of the represented argumentation framework.

Definition 12. A second order argumentation framework is a tuple $\langle \mathcal{A}_C, \overline{\mathcal{A}}, \text{not}, \mathcal{A}_\#, \# \rangle$, where \mathcal{A}_C is a set of coalition arguments, $\overline{\mathcal{A}}$ is a set of arguments such that $|\overline{\mathcal{A}}| = |\mathcal{A}_C|$, not is a bijection from \mathcal{A} to $\overline{\mathcal{A}}$, $\mathcal{A}_\#$ is a set of arguments that coalitions attack each other, and $\# \subseteq (\mathcal{A}_C \times \overline{\mathcal{A}}) \cup (\overline{\mathcal{A}} \times \mathcal{A}_\#) \cup (\mathcal{A}_\# \times \mathcal{A}_C) \cup (\mathcal{A}_C \times \mathcal{A}_\#)$ is a binary relation on the set of arguments such that for $a \in \mathcal{A}_C$ and $b \in \overline{\mathcal{A}}$ we have $a \# b$ if and only if $b = \text{not}(a)$, and for each $a \in \mathcal{A}_\#$, there is precisely one $b \in \overline{\mathcal{A}}$ such that $b \# a$ and precisely one $c \in \mathcal{A}_C$ such that $a \# c$. A second order argumentation framework $\langle \mathcal{A}_C, \overline{\mathcal{A}}, \mathcal{A}_\#, \# \rangle$ represents $\langle \mathcal{A}, \# \rangle$ if and only if $\mathcal{A} = \mathcal{A}_C \cup \overline{\mathcal{A}} \cup \mathcal{A}_\#$. The extensions of $\langle \mathcal{A}_C, \overline{\mathcal{A}}, \mathcal{A}_\#, \# \rangle$ are the extensions of the represented argumentation framework.

The intuition behind second order argumentation is the following. Attack relations are substituted by arguments $\mathcal{A}_\#$ representing attacks, so that these can be attacked in turn by further arguments. However, the arguments $\mathcal{A}_\#$ differently from the original attack relations have an existence which is more independent from the arguments which they stem from. E.g., the attack of C_1 to C_2 is substituted by an argument $C_{1,2}$. If C_1 is attacked, then its attack relations should not be considered anymore. Instead, if the attack

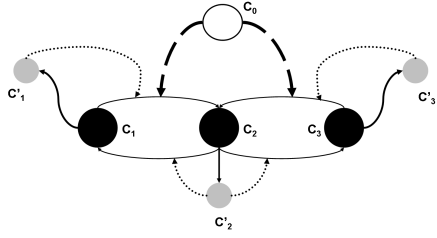


Fig. 3. Graphical representation of Example 7

relation is substituted by an argument in $\mathcal{A}_\#$, this argument continues to attack other arguments even when the argument it stems from is attacked: e.g., if C_1 is attacked, $C_{1,2}$ continues to attack C_2 . Thus, to avoid this problem, for each argument in \mathcal{A}_C a new argument C'_1 in $\overline{\mathcal{A}}$ is created. This argument is created such that it attacks all the arguments of $\mathcal{A}_\#$ representing the attack relations of C_1 against other arguments. Besides C_1 , C'_1 is added, attacking $C_{1,2}$. C'_1 however should not attack $C_{1,2}$ at any moment, but only when C_1 is attacked: for this reason, an attack relation between C_1 and C'_1 is added, so that C'_1 attacks the attack relations stemming from C_1 only when C_1 is attacked.

The following example illustrates the second order argumentation theory. The main feature of $not(C)$ arguments is just to ensure that if an argument is not accepted, then it cannot attack other arguments. The argument C_0 is a dummy argument to represent the preferences, here used to map the second order framework to the preference-based one. This example is visualized in Figure 3.

Example 8 (Continued). Let $\langle \mathcal{A}_C, \overline{\mathcal{A}}, not, \mathcal{A}_\#, \# \rangle$ be a second order argumentation framework, where $\mathcal{A}_C = \{C_1, C_2, C_3, C_0\}$ is a set of coalition arguments, $\overline{\mathcal{A}} = \{C'_1, C'_2, C'_3, C'_0\}$, not is the bijection $not(C_i) = C'_i$, $\mathcal{A}_\# = \{C_{1,2}, C_{2,1}, C_{2,3}, C_{3,2}\}$ is a set of arguments that coalitions attack each other, and

$$\{C_1 \# C'_1, C_2 \# C'_2, C_3 \# C'_3, C_0 \# C'_0, C'_1 \# C_{1,2}, C'_2 \# C_{2,1}, C'_2 \# C_{2,3}, C'_3 \# C_{3,2}, \\ C_{1,2} \# C_2, C_{2,1} \# C_1, C_{2,3} \# C_3, C_{3,2} \# C_2, C_0 \# C_{1,2}, C_0 \# C_{3,2}\}$$

is a binary relation on the set of arguments. For the nested attack relations, we also write $C_0 \# (C_1 \# C_2)$ and $C_0 \# (C_3 \# C_2)$. The acceptable argument is C_2 , together with C_0 , C'_1 , C'_3 , $C_{2,1}$, $C_{2,3}$, for any kind of semantics. We can visualize second order argumentation frameworks by not visualizing $\overline{\mathcal{A}}$ or $\mathcal{A}_\#$, and visualizing an indirect attack from an element of \mathcal{A}_C to \mathcal{A}_C via an element of $\mathcal{A}_\#$ as an arrow, and an attack of an element of \mathcal{A}_C to an element of $\mathcal{A}_\#$ as an attack on an attack relation, see [18] for examples of such a visualization. This example shows that arguments that attack attack relations do that directly.

Example 9 (Continues Example 5 - See Figure 2). Assume instead that $dep(a, \{d\}, \{\{g_4\}\})$ is not present since the beginning and it happens that agent g of C_2 has the power to create it: i.e., it is substituted by $dyndep(a, \{d\}, \{g\}, \{\{g_4\}\})$. Thus, C_2 attacks

the attack relation between C_1 and C_2 , $C_2 \# (C_1 \# C_2)$ by Definition 8: if coalition C_1 remains potential, since nothing guarantees that g will create goal g_4 of agent a without receiving anything in exchange, then it cannot attack any other coalition. Moreover, assume that $dep(e, \{f\}, \{\{g_6\}\})$ is not present since the beginning and it happens that agent h of C_2 has the power to create it and, thus, the dependency is substituted by $dyndep(e, \{f\}, \{h\}, \{\{g_6\}\})$. Thus, C_2 attacks the attack relation between C_2 and C_3 , $C_2 \# (C_3 \# C_2)$ by Definition 8. The only extension is $\{C_2\}$.

We illustrate this situation in Figure 2: the attack relation on attack relations is depicted as bold dashed arrows pointing on other arrows.

Note that if in Example 8 argument C'_0 is identified with C_2 (and C'_0 with C'_2), a second order argumentation framework for the current example is obtained.

Finally, we show how to relate the argumentation frameworks. We illustrate how second order argumentation frameworks can be seen as an extension of Dung's argumentation framework.

Proposition 2. *An argumentation framework $\langle \mathcal{A}, \#_1 \rangle$ represents a second order argumentation framework $\langle \mathcal{A}_C, \bar{\mathcal{A}}, not, \mathcal{A}_\#, \#_2 \rangle$ when*

1. $\mathcal{A}_C = \mathcal{A}$, and
2. there is an element $a \in \mathcal{A}_\#$ for each pair of arguments $b, c \in \mathcal{A}$ such that $b \#_1 c$, with $not(b) \#_2 a$ and $a \#_2 c$.
3. there are no arguments $a \in \mathcal{A}$ and $b \in \mathcal{A}_\#$ such that $a \#_2 b$.

If $\langle \mathcal{A}, \#_1 \rangle$ represents $\langle \mathcal{A}_C, \bar{\mathcal{A}}, not, \mathcal{A}_\#, \#_2 \rangle$, then the extensions of $\langle \mathcal{A}, \#_1 \rangle$ correspond to the extensions of $\langle \mathcal{A}_C, \bar{\mathcal{A}}, not, \mathcal{A}_\#, \#_2 \rangle$ intersected with \mathcal{A} .

4 Related Work

Sichman [22] presents coalition formation using a dependence-based approach based on the notion of social dependence introduced by Castelfranchi [12].

The application of argumentation frameworks to coalition formation has been discussed by Amgoud [1] and by Bulling et al. [10]. In Amgoud's paper, a coalition may have a cost and a profit, so the agents are able to evaluate each coalition. Unlike Amgoud's work [1], we do not provide the present paper with a proof theory since it is derivable from the argumentation theory's literature. Moreover, unlike Amgoud's paper [1], we do not define any kind of dialogue model among the agents involved in coalitions.

Another formal approach to reason about coalitions is coalition logic [19] and Alternating Temporal Logic (ATL), describing how a group of agents can achieve a set of goals, without considering the internal structure of the group of agents [2][5][16]. See [21] for a further discussion. Bulling et al. [10], instead, combine the argumentation framework and ATL with the aim to develop a logic through which reasoning at the same time about abilities of coalitions of agents and about coalitions formation. They provide a formal extension of ATL in which the actual computation of the coalition is modeled in terms of argumentation semantics. The key construct in ATL expresses that a coalition of agents can enforce a given formula. [10] presents a first approach towards extending ATL for modeling coalitions through argumentation. A difference regarding Amgoud's paper, is the intuition, in accordance with ATL, where larger

coalitions are more powerful than smaller ones. In Bulling's paper, the actual computation of the coalition is modeled in terms of a given argumentation semantics in the context of coalition formation. The paper's approach is a generalization of the framework of Dung for argumentation, extended with a preference relation. The basic notion is that of a coalitional framework containing a set of elements (usually represented as agents or coalitions), an attack relation (for modeling conflicts among these elements), and a preference relation between these elements (to describe favorite agents/coalitions). The notion of coalitional framework is based on the notion of framework for generating coalition structures presented in Amgoud's paper.

5 Summary and Further Research

Frameworks for arguing about coalitions are based on non-monotonic logic and are therefore formal and abstract, whereas social theories about agent coalitions typically are based on conceptual modeling languages and therefore semi-formal and detailed. In this paper we bridge the gap between these two research areas such that social viewpoints can be used to argue about coalitions.

For arguing about coalitions, we define three social viewpoints with abstraction and refinement relations between them, and adapt existing coalition argumentation theory to reason about the coalitions defined in the most abstract viewpoint, the coalition view representing a coalition as a set of dynamic dependencies between agents. We define dynamic dependence networks by making the dependence relation conditional to the agents that have the power to create it, distinguishing two kinds of power, not only to fulfill goals as in static networks but also to create dependencies. Coalitions are defined by a kind of "contracts" in which each agent both contributes to the coalition, and profits from it.

We need to use different viewpoints to argue about coalitions since dynamic dependencies underline two different aspects of coalitions. From an internal point of view, the agents inside a coalition can be described by viewpoints and thus we represent the coalition, depending on the adopted abstraction, as a set of agents with their goals and skills or as a set of agents related due the notion of power or, finally, as a set of dynamic dependencies. From an external point of view, instead, the addition of a new dependence can be represented as an attack relation from a coalition to another one at coalitional view level.

Subjects of further research are the use of our new theory for coalition formation. For example, when two agents can make the other depend on itself and thus create a potential coalition, when will they do so? Do these new ways to create coalitions make the system more efficient, or more convivial? Moreover, new measures have to be defined for the dynamic dependence networks, where we may find inspiration in dynamic social network analysis.

References

1. Amgoud, L.: An Argumentation-Based Model for Reasoning About Coalition Structures. In: Proceedings of ArgMAS 2005, pp. 217–228 (2005)
2. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. *Journal of ACM* 49(5), 672–713 (2002)

3. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. *Artif. Intell.* 171(10-15), 619–641 (2007)
4. Boella, G., Sauro, L., van der Torre, L.: From social power to social importance. In: *Web Intelligence and Agent Systems*, pp. 393–404. IOS Press, Amsterdam (2007)
5. Boella, G., Sauro, L., van der Torre, L.: Social Viewpoints on multiagent Systems. In: *Proceedings of AAMAS 2004*, pp. 1358–1359 (2004)
6. Boella, G., Sauro, L., van der Torre, L.: Strengthening Admissible Coalitions. In: *Proceedings of ECAI 2006*, pp. 195–199 (2006)
7. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems Journal* 8, 203–236 (2004)
8. Brewka, G.: Reasoning about Priorities in Default Logic. In: *AAAI 1994*, pp. 940–945 (1994)
9. Broersen, J., Dastani, M., Hulstijn, J., van der Torre, L.: Goal generation in the BOID architecture. *Cognitive Science Quarterly* 2(3-4), 428–447 (2002)
10. Bulling, N., Chesnevar, C.I., Dix, J.: Modelling Coalitions: ATL + Argumentation. In: *Proceedings of AAMAS 2008*, pp. 681–688 (2008)
11. Caire, P., Villata, S., van der Torre, L., Boella, G.: Conviviality Masks in Role-Based Institutions Multi-Agent Teleconferencing in Virtual Worlds. In: *Proceedings of AAMAS 2008*, pp. 1265–1268 (2008)
12. Castelfranchi, C.: The micro-macro constitution of power. *Protosociology* 18, 208–269 (2003)
13. 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)
14. Hindriks, K.V., De Boer, F.S., Van der Hoek, W., Meyer, J.-J.C.: Agent Programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 357–401 (1999)
15. van der Hoek, W., Wooldridge, M.: Tractable Multiagent Planning for Epistemic Goals. In: *Proceedings of AAMAS 2002*, pp. 1167–1174 (2002)
16. van der Hoek, W., Jamroga, W., Wooldridge, M.: A Logic for Strategic Reasoning. In: *Proceedings of AAMAS 2005*, pp. 157–164 (2005)
17. Kaci, S., van der Torre, L.W.N., Weydert, E.: On the Acceptability of Incompatible Arguments. In: Mellouli, K. (ed.) *ECSQARU 2007*. LNCS, vol. 4724, pp. 247–258. Springer, Heidelberg (2007)
18. Modgil, S.: An abstract theory of argumentation that accommodates defeasible reasoning about preferences. In: Mellouli, K. (ed.) *ECSQARU 2007*. LNCS, vol. 4724, pp. 648–659. Springer, Heidelberg (2007)
19. Pauly, M.: A Modal Logic for Coalitional Power in Games. *Journal of Logic and Computation* 12, 146–166 (2002)
20. Prakken, H., Sartor, G.: Argument-Based Extended Logic Programming with Defeasible Priorities. *Journal of Applied Non-Classical Logics* 7(1), 25–75 (1997)
21. Sauro, L.: Formalizing admissibility criteria in coalition formation among goal directed agents, Dipartimento di Informatica, Università di Torino, PhD Thesis (2005)
22. Sichman, J.S.: DEPINT: Dependence-Based Coalition Formation in an Open Multi-Agent Scenario. *Journal of Artificial Societies and Social Simulation* 1(2) (1998)
23. Sichman, J.S., Conte, R.: Multi-agent dependence by dependence graphs. In: *Proceedings of AAMAS 2002*, pp. 483–490 (2002)

Changing Institutional Goals and Beliefs of Autonomous Agents

Guido Boella¹, Leendert van der Torre², and Serena Villata¹

¹ Dipartimento di Informatica, University of Turin, Italy

² Computer Science and Communication, University of Luxembourg, Luxembourg

Abstract. Agents are autonomous and thus their goals and beliefs cannot be changed by other agents or the environment, but from outside the agents we can change their *institutional* goals and beliefs, that is, the responsibilities and powers associated with their roles. In this paper, we introduce a model of institutional dynamics, where the dynamics of an institution is modeled by the creation or removal of responsibilities and powers. We illustrate the change of institutional goals and beliefs using a government scenario.

1 Introduction

Institutions are structures and mechanisms of social order and cooperation governing the behavior of a set of individuals. However, the formal analysis of the institutions is challenging due to the complexity of its dynamics. For example, the institution itself may change over time due to the behavior of the agents. To model the dynamics of the institution, we use dependence networks developed by Sichman and Conte [11], as they are used in the early requirement analysis of Tropos [6].

The research question of this paper is: How to model the dynamics of dependence networks between agents due to *institutional* change? In our model presented in [3] we distinguish four views on multiagent systems without making institutions explicit. The agent view represents the agents of the systems introducing conditional goals, such as those goals that can be added by an agent to another one. The power view is presented as an abstraction of the agent view and it introduces two functions associating the agents with the goals they can create to agents and with the goals they can achieve. The dynamic dependence view is defined as an abstraction of the power view. In the model of [3], dependencies can be created by a set of agents making sets of agents dependent on each other, thanks to the power to trigger conditional goals. In this paper we present an extension of this model with also removal of dependencies. The challenge of the present paper is to explain how on the one hand agents are autonomous and, thus, their goals and beliefs cannot change, but on the other hand, the dependences between agents can change in an institution.

The paper is organized as follows. Section 2 introduces the running example, Section 3 and 4 introduce our model of institutions, and Section 5 introduces the dynamic social network. Related work and conclusions end the paper.

2 Running Example: The Government Scenario

The government scenario models the relations among ministers, where the prime minister has more powers than the other ministers, who have powers according to their domain. At the non-institutional level, a minister needs a ministerial car if she has to travel in town, she has to ask to the suitable office if she needs a translation service, she has to ask to the office of public relations to set a press conference if she needs to release a press statement, she has to contact the office with the job to update web site and to ask it to do the changes if she needs the publication on the web site of a particular office of new information, and so on. Each item can be modeled as a goal of the minister and the agents on which she depends to achieve her goals. For example, she depends on the office of public relations to achieve the goal to program a press conference. Together, these dependencies can be modeled as a social network.

From an institutional point of view, the government structure is hierarchical, in the sense that the Prime Minister has more powers than all the other ministers, and the ministers have more powers than deputy-ministers. Consequently, the ministers depend on the prime minister. If the Prime Minister delegates to the foreign secretary some diplomatic topic, then all the other ministers and the secretaries have to refer to the foreign secretary regarding this topic. Likewise, the Prime Minister can also remove a delegated topic again, for example if there is a case of uncorrect behavior, taking her delegations *ad interim*. From this moment, it will be the Prime Minister to which the other ministers have to refer. Ministers also depend on each other. These institutional dependencies among ministers is typically given by authorizations. For example, the minister of transport needs an authorization from the minister of infrastructures to bridge a river, or the minister of public works needs funding from the minister of finance to call for tenders to build the bridge. These examples show how a minister $M1$ having a goal $G1$ depends on minister $M2$ to achieve it to have the authorization.

The institutional powers of the Prime Minister can create new dependencies among ministers. His institutional powers are that he can give a permission to the other ministers to do something, for example the permission to be absent to a council of ministers for serious reasons, and he can create obligations to other ministers, for example, the Prime Minister can oblige the minister of transports to present a document within a precise date. If the Prime Minister creates such an obligation, then he may create also a dependency of the minister to other ministers to obtain the document. Moreover, consider the example in which the minister of Transports depends on the minister of Infrastructure to have the authorization to start to build a new road. If the minister of Infrastructure is under investigation for corruption and then he is removed from his role of minister, then the Prime Minister can take the office of the minister of Infrastructure and so all his powers. In this case, the minister of Transports depends on the Prime Minister to obtain his authorization to start building the road.

3 Social Viewpoints for Dynamic Dependence Networks

In our model introduced in [3], four views are defined to describe a multiagent system: the agent view, the power view, the dynamic dependence view and the coalition view. While the agent view represented the agents of the systems, their goals and the actions that they can perform to achieve these goals, the power view, based on the concept of social power introduced by Castelfranchi [8], introduced conditional goals, such as those goals that can be added by an agent to another one. We defined the dynamic dependence view as an abstraction of the power view, represented by a set of dependencies that can be added thanks to existence of conditional goals. Abstracting from the dynamic dependence view, we defined the coalition view representing a coalition as a set of dynamic dependencies where each agent either creates a dependency or fulfills a goal.

In this model, the power to trigger a goal is distinguished from the power to fulfill a goal. Due to the power to create goals, dependence relations are no longer static, but they can be created by agents. We extended the dependence networks developed by Conte and Sichman [11], following our previous results in [7]. In the abstraction from the power view to a dynamic dependence network, the creation of a dynamic dependency is based only on the power to create goals. In other models, creating a dependency can also be due to creation of new skills of agents. For formal details, see [3].

These four views can be applied to our running example since every minister has a set of private goals, beliefs and abilities and, according to the actions he can perform he has the power to see to a number of goals, his or of the other agents. Conditional goals are represented in our scenario, for example, by the decision of the Prime Minister to give to the minister of transports to present a document within a precise date. In this case, the Prime Minister has the power to trigger this particular goal. These conditional goals are the base of the change inside the dependence networks, transforming it in a dynamic dependence network.

4 The Institutional View

A social structure is modeled as a collection of agents, playing roles regulated by norms where “interactions are clearly identified and localized in the definition of the role itself” [13]. The notion of role is notable in many fields of Artificial Intelligence and, particularly, in multiagent systems where the role is viewed as an instance to be adjoined to the entities which play the role.

The institutional view is defined as follows:

Definition 1 (Institutional view (IV)). $IV = \langle RL, IF, RG, X, igoals: RL \rightarrow 2^{RG}, iskills: RL \rightarrow 2^X, irules: 2^X \rightarrow 2^{IF} \rangle$ consists of a set of role instances RL , a set of institutional facts IF , a set of public goals attributed to roles, a set of actions X , a function $igoals$ that relates with each role the set of public goals it is committed to, a function $isksills$ that describes the actions each role can perform, and a set of institutional rules $irules$ that relates a set of actions and the set of institutional facts they see to.

The institutional view assigns to each participant a set of public goals, describing what he can do, e.g. authorize to built a bridge, and should do, e.g. be present at a council of ministers. Our scenario allows to enforce the behavior of the agents in the institution, for example, by blocking them from making statements contradicting facts, or by performing forbidden (virtual) actions, such as e.g. embezzle public money.

The social reality is provided with two distinct views, the material one, called the agent view in [3], and the institutional one that aims to regulate the behaviour of the agents. In a multiagent system each agent has a set of facts and goals that the other agents cannot change since agents are autonomous, formally presented in the agent view. Thanks to its existence inside a social structure, to each agent is added also new sets of facts and goals called the institutional ones and that can be viewed and also modified by the other agents as regards their institutional role.

The agents start with their sets of personal goals and beliefs and, only after their insertion inside a particular social structure they enlarge their sets of goals and beliefs. In particular, the set of goals is enlarged with new normative goals that represent the responsibilities of the agent inside its social structure while the set of beliefs is enlarged with new normative beliefs representing the set of constitutive norms of the systems, norms based on the collective acceptance of the society representable by means of an institutional ontology.

An *Institutional Social Network* is a social network representing set of individuals regulated by norms and containing the application of social roles to each individual involved.

5 Institutional Dynamic Dependence Networks

The passage from one institutional view to another one can be viewed as a dynamic social dependence network composed by all the social dependence networks coupled with the different institutional views. The main changes, that can occur to the institutional view to make it dynamic and pass from an institutional view to another one, are the addition or deletion of an *igoal*, of an *iskill* and of an *irule*. These additions and deletions change the number of dependencies and what agents are involved in them, passing from a social dependence network to another one. This change can be represented by means of dynamic social dependence networks. We extend the definition of dynamic dependence network proposed in [3] with the possibility not only to add dependencies between agents but also to remove them. Dynamic dependence networks are defined as follows:

Definition 2 (Dynamic Social Dependence Networks (DDN)). A *dynamic social dependence network* is a tuple $\langle A, G, \text{dyndep}^-, \text{dyndep}^+ \rangle$ where:

- A is a set of agents and G is a set of goals.
- $\text{dyndep}^- : A \times 2^A \times 2^A \rightarrow 2^{2^G}$ is a function that relates with each triple of an agent and two sets of agents all the sets of goals in which the first depends on the second, unless the third deletes the dependency. The static dependencies are defined by $\text{dep}(a, B) = \text{dyndep}^-(a, B, \emptyset)$.

- $dyndep^+ : A \times 2^A \times 2^A \rightarrow 2^{2^G}$ is a function that relates with each triple of a agent and two sets of agents all the sets of goals on which the first depends on the second, if the third creates the dependency.

The following definition introduces measures for existing dependencies and achieved goals. The overall value assigned to each dependence network is the ratio among the total number of goals that can be achieved by the agents of the network and the total number of dependencies among these agents.

Definition 3 (Measures)

- Number of goals that are achieved inside the network.
 - $nsg : |S|$ where $S = \{G_1, G_2, \dots, G_n\} \forall G_{i=1, \dots, n} \in dep(A_1, A_2, G_i)$
- Number of dependencies inside the network.
 - $nd : |D|$ where $D = \{dep_1, dep_2, \dots, dep_n\} \forall dep_{i=1, \dots, n} \in DN = \langle A, G, dep_i \rangle$.
- The goals-dependencies ratio is defined as:
 - $GD - ratio = \frac{nsg}{nd}$ with $nsg \leq nd$.

Since a dynamic social dependence network is a superimposition of a number of dependence networks, we define a $\Delta_{GD-ratio}$ that expresses how much changes the ratio between the number of achieved goals and the number of dependencies among the different dependence networks composing a dynamic dependence network.

Example [I](#) illustrates the addition of an *irule* and of an *iskill* by role instance Pm (Prime Minister). In the first case, the Prime Minister adds a new *irule*, that joins the institutional action ix_N with the institutional goal pg_N , that it allows to achieve. Since the minister able to perform the institutional action ix_N is the minister of transport (Tm) and the two ministers, infrastructure (Im) and finance (Fm), have the institutional goal pg_N , there is the setting of two new dynamic dependencies. In the second case, the Prime Minister adds a new institutional *iskill* ix_e to the minister of finance that allows the creation of a new dynamic dependence in which the Prime Minister depends on the minister of finance to achieve the institutional goal pg_5 , allowed by ix_e .

Example 1. Consider the following additions:

- Role Pm adds to the institutional view a new *irule*: $IV \rightarrow IV + \{irules \cup \{\{ix_N\}, \{pg_N\}\}\}$ where $\{pg_N\} \subseteq igoals(Im)$ and $\{pg_N\} \subseteq igoals(Fm)$ and $\{ix_N\} \subseteq iskills(Tm)$ then $dep_2 = dep + dyndep^+(Im, Tm, Pm, \{pg_N\}) + dyndep^+(Fm, Tm, Pm, \{pg_N\})$ and $nsg_2 = nsg_1 + 1$ and $nd_2 = nd + 2$;
- Role Pm adds to the institutional view also a new *iskill*: $IV \rightarrow IV + \{isksills \cup \{Fm, \{ix_e\}\}\}$ where $irules(K, \{pg_5\})$ and $ability(\{pg_5\}) = \{ix_e\}$ such that $\{pg_5\} \subseteq K$ and $\{pg_5\} \subseteq igoals(Pm)$ then $dep_2 = dep + dyndep^+(Pm, Fm, Pm, \{pg_5\})$ and $nsg_2 = nsg + 1$ and $nd_2 = nd + 1$;

Example 2 represents the application of the two additions on a dynamic social dependence network, as shown in Figure [II](#)

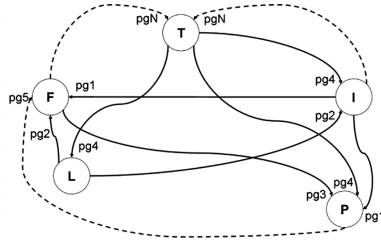


Fig. 1. DDN of Example 2 where dotted arcs represent new dependencies

Example 2. Consider the following dynamic social dependence network:

1. Agents $A = \{T, I, F, L, P\}$ and Goals $G = \{pg_1, pg_2, pg_3, pg_4, pg_5, pg_N\}$;
2.
 - $dep(\{I\}, \{P\}) = \{\{pg_1\}\}$: agent I depends on agent P to achieve goal $\{pg_1\}$;
 - $dep(\{T\}, \{I\}) = \{\{pg_4\}\}$: agent T depends on agent I to achieve goal $\{pg_4\}$;
 - $dep(\{L\}, \{I, F\}) = \{\{pg_2\}\}$: agent L depends on agents I, F to achieve goal $\{pg_2\}$;
 - $dep(\{T\}, \{P, L\}) = \{\{pg_4\}\}$: agent T depends on agents P, L to achieve goal $\{pg_4\}$;
 - $dep(\{I\}, \{F\}) = \{\{pg_1\}\}$: agent I depends on agent F to achieve goal $\{pg_1\}$;
 - $dep(\{F\}, \{P\}) = \{\{pg_3\}\}$: agent F depends on agent P to achieve goal $\{pg_3\}$;
 - $dyndep^+(\{I, F\}, \{T\}, \{P\}) = \{\{pg_N\}\}$: agents I, F depend on agent T to achieve goal $\{pg_N\}$ if it is created by agent P ;
 - $dyndep^+(\{P\}, \{F\}, \{P\}) = \{\{pg_5\}\}$: agent P depends on agent F to achieve goal $\{pg_5\}$ if it is created by agent P ;
3. The following measures show that, thanks to an increase of the number of dependencies, also the number of achieved goals increases: $GD - ratio_1 = \frac{4}{8}$ and $GD - ratio_2 = \frac{6}{11}$ with $nsg \leq nd$;

6 Related Work

Sierra [12] introduces Electronic Institutions (EIs) providing the virtual analogue of human organizations in which agents, playing different organizational roles, interact to accomplish individual and organizational goals. Roles are defined as patterns of behavior and the purpose of their normative rules is to affect the behavior of agents by imposing obligations or prohibitions. Another approach to EIs is given by [4]. They propose the use of 3D Virtual Worlds where an institution is represented as a building where the participants are represented as avatars and once they enter the building their actions are validated against the specified institutional rules. The problem of dynamic institutions is treated

in [5] as an extension to EIs definition with the capability to decide in an autonomous way how to answer dynamically to changing circumstances through norm adaptation and changes in institutional agents.

As originally defined, dependence networks lack two ingredients: a normative structure and a dynamic representation of networks of social structures. Normative multiagent systems provide agents with abilities to automatically devise societies coordinating their behavior via obligations, norms and social laws [2]. The definition of power of Boella [1] can be directly applied to the description of the institutional view. Also the ability to achieve goals can be directly defined in terms of facts, skills and goals attributed to roles following the definition given in [1]. The presented formal model can be extended with obligations, as in [2]. Dependencies due to norms like obligations and permissions can be modeled by means of social dependence networks as in [7], however, institutional powers cannot be captured by the existing dependence networks formalism, since they introduce a dynamic element. By exercising a power, an agent transforms a social dependence structure into a new one by adding or removing dependencies at the institutional level of the social structure. Thus, in our paper, power is seen as the base of the change differently from what expresses by Jones and Sergot [10] and Grossi [9].

7 Conclusions

In this paper we show how to model the dynamics of dependence networks between agents due to institutional change, where the institutions are used to enforce the global behaviour of the society and to assure that the global goals of the society are met. Roughly, the behavior of the agents leads to a set of institutional facts, leading to the addition and removal of responsibilities and powers associated with the roles. This change of the institutional goals and beliefs of agents constitutes the dynamic behavior of the institution. The challenge of this paper is to explain how on the one hand, agents are autonomous and, thus, their goals and beliefs cannot change, but on the other hand, the dependences between agents can change in an institution. This challenge is twofold. First, we explain how we can change their institutional goals and beliefs, that is, the responsibilities and powers associated with their roles. Second, we explain how institutional goals are distinguished from the private goals and beliefs.

The change of institutional goals and beliefs of the agents is explained, first, by an informal example based on an hypothetical government. Second, it is explained by additions and deletions of dependencies between the agents using dependence networks developed by Sichman and Conte [11]. Moreover, we define measures to analyze dependence networks. As illustrated by the government scenario, the uniform combined model of role playing agents with private and institutional powers and goals provides an intuitive representation for dynamics in terms of modification of the institution, and the network measures are used to analyze this dynamics.

The distinction between changing institutional goals and beliefs and changing private goals and beliefs is based on the notion of institutional power. We introduce the notion of institution based on the previously defined model of cognitive agent of [3], associating to each agent a role instance with its institutional goals, beliefs and capabilities. In [3] there is no definition of institution and the changes are governed by power and are represented by conditional goals, so in this paper we present an extension to the definition of dynamic dependence network of [3] with also removal of dependencies. Due to the concept of institutional power, the present paper shows the possibility to change the institutional goals and beliefs of the agents maintaining agents' autonomy.

References

1. Boella, G., Sauro, L., van der Torre, L.: From social power to social importance. In: *Web Intelligence and Agent Systems*, pp. 393–404. IOS Press, Amsterdam (2007)
2. Boella, G., van der Torre, L.: Power in Norm Negotiation. In: Nguyen, N.T., Grzech, A., Howlett, R.J., Jain, L.C. (eds.) *KES-AMSTA 2007*. LNCS, vol. 4496, pp. 436–446. Springer, Heidelberg (2007)
3. Boella, G., van der Torre, L., Villata, S.: Social Viewpoints for Arguing about Coalitions. In: *Proceedings of PRIMA 2008*. LNCS, vol. 5357. Springer, Heidelberg (2008)
4. Bogdanovych, A., Esteva, M., Simoff, S., Sierra, C., Berger, H.: A Methodology for Developing multiagent Systems as 3D Electronic Institutions. In: Luck, M., Padgham, L. (eds.) *Agent-Oriented Software Engineering VIII*. LNCS, vol. 4951, pp. 103–117. Springer, Heidelberg (2008)
5. Bou, E., Lopez-Sanchez, M., Rodriguez-Aguilar, J.A.: Adaptation of Automatic Electronic Institutions Through Norms and Institutional Agents. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) *ESAW 2006*. LNCS, vol. 4457, pp. 300–319. Springer, Heidelberg (2007)
6. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems Journal* 8, 203–236 (2004)
7. Caire, P., Villata, S., van der Torre, L., Boella, G.: Conviviality Masks in Role-Based Institutions Multi-Agent Teleconferencing in Virtual Worlds. In: *Proceedings of AAMAS 2008*, pp. 1265–1268 (2008)
8. Castelfranchi, C.: The micro-macro constitution of power. *Protosociology* 18, 208–269 (2003)
9. Grossi, D.: *Designing Invisible Handcuffs: Formal Investigations in Institutions and Organizations for Multi-agent Systems*, PhD Thesis, SIKS Dissertation Series 2007-16 (2007)
10. Jones, A.J.I., Sergot, M.: A Formal Characterization of Institutionalised Power. *Logic Journal of IGPL* (2003)
11. Sichman, J.S., Conte, R.: Multi-agent dependence by dependence graphs. In: *Proceedings of AAMAS 2002*, pp. 483–490 (2002)
12. Sierra, C., Rodriguez-Aguilar, J.A., Noriega, P., Arcos, J.L., Esteva, M.: Engineering multi-agent systems as electronic institutions. *European Journal for the Informatics Professional* 5, 33–39 (2004)
13. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: The Gaia methodology. *IEEE Transactions of Software Engineering and Methodology* 12, 317–370 (2003)

Reasoning about Constitutive Norms, Counts-As Conditionals, Institutions, Deadlines and Violations

Guido Boella¹, Jan Broersen², and Leendert van der Torre³

¹ University of Torino, Italy

² University of Utrecht, The Netherlands

³ Computer Science and Communication, University of Luxembourg, Luxembourg

Abstract. Reasoning about norms and time is of central concern to the regulation or control of the behavior of a multi-agent system. In earlier work we introduce a representation of normative systems that distinguishes between norms and the detached obligations of agents over time. In this paper we consider constitutive norms and the detached counts-as conditionals and institutional facts in this framework, we introduce deadlines in the regulative norms, and we consider the corresponding role of violations. We focus on the reasoning tasks to determine whether a constitutive or regulative norm is redundant in a normative system and whether two normative systems are equivalent. We distinguish counts-as equivalence, institutional equivalence, obligation equivalence and violation equivalence, depending on whether we are interested in all normative consequences, or only a subset of them. For the various notions of equivalence, we give sound and complete characterizations.

1 Introduction and Running Example

Reasoning about norms and time is of central concern to institutions [1] for the regulation or control of the behavior of a multiagent system [2,3]. Institutions are normative systems consisting of regulative norms like obligations, prohibitions and permissions, and constitutive norms like “X counts as Y in context C” [4,5,6,7]. For example, in the context of a selling contract, an electronic signature of agent John j counts as his signature, and his signature counts as the fact that the signer j owes the specified amount of money to the seller Peter p . The constitutive norms are used to derive institutional facts, also called intermediate concepts, that there is a signature, or that one agent owes another agent money, which is then used to derive obligations and permissions using regulative norms. For example, if agent j owes agent p money, then he has to pay him before a deadline, and if j has paid p , then p has to give j a receipt before another deadline.

Our running example is normative system $NS = \langle CN, RN \rangle$ with constitutive norms $CN = \{(\text{esign}_{jp} \text{ counts-as } \text{sign}_{jp} \text{ in } \text{contr}), (\text{sign}_{jp} \text{ counts-as } \text{owe}_{jp} \text{ in } \text{contr})\}$ and regulative norms $RN = \{(\text{owe}_{jp}, \text{pay}_{jp}, d), (\text{pay}_{jp}, \text{receipt}_{pj}, e)\}$, where the latter are read as “if owe_{jp} then obligatory pay_{jp} before d ”, and “if pay_{jp} then obligatory receipt_{pj} before e ”. The norms are used to label temporal structures like the one in Figure 1, which must be read as follows. A circle visualizes a node of the tree and thus a moment in time, and the words within the circle visualize the brute facts which hold at that node.

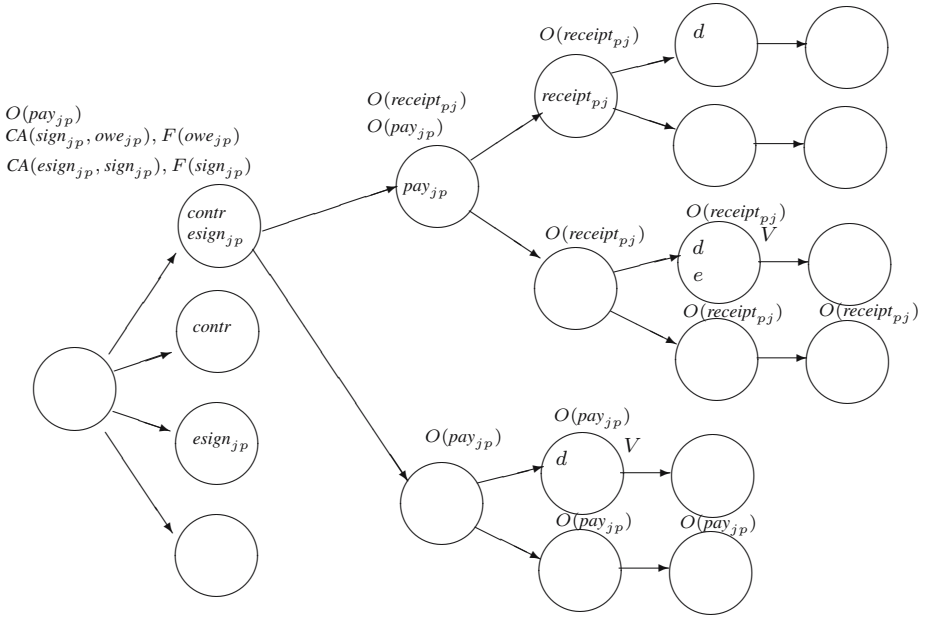


Fig. 1. Labeling of the temporal structure using the normative system $NS = \langle CN, RN \rangle$ with $CN = \{(esign_{jp}, sign_{jp}, contr), (sign_{jp}, owe_{jp}, contr)\}$ and $RN = \{(owe_{jp}, pay_{jp}, d), (pay_{jp}, receipt_{pj}, e)\}$

An arrow from one node to another visualizes that the latter node is among the possible future moments of the former. The root of the tree is visualized on the left hand side, and four distinct nodes can be reached from it. In all these nodes we have or do not have *contr*, and we have or do not have *esign_{jp}*. For all these four nodes we can reach later nodes in the paths, visualized on the right hand side of the figure, in which we no longer have *contr* or *esign_{jp}*, but the agents have performed actions *pay_{jp}* or *receipt_{pj}*, or the deadlines *d* and *e* expire. In that case it is possible that an obligation is violated (*V*).

Since there are various subtle choices to be made in the way institutional facts and obligations are derived from a normative system, we study ways to reason about the redundancy and equivalence of such systems. Makinson and van der Torre introduce the input/output logic framework to reason about regulative norms and obligations, and Broersen and van der Torre [8,9] extend it to reason about obligations in time. However, normative systems contain more than just regulative norms, and in this paper we therefore consider constitutive norms in this temporal framework. Just like regulative norms are used to detach obligations, constitutive norms are used to detach counts-as conditionals and institutional facts. In particular, we address the following questions, not addressed in our earlier papers [8,9] on this approach:

1. How to label temporal structures with counts-as conditionals and institutional facts, given a set of constitutive norms?
2. When are constitutive norms redundant, and when are constitutive norm systems equivalent?

3. How to extend the labeling of temporal structures with persistent regulative norms with deadlines?
4. When are persistent regulative norms with deadlines redundant, and when are regulative norm systems equivalent?
5. How to label temporal structures with violations, and how to define violation redundancy and equivalence?

As a methodology we use input/output logic to represent both regulative and constitutive norms, due to their common conditional character, and we introduce two distinctions: on the one hand, we distinguish between regulative norms and obligations, and, on the other hand, between constitutive norms and counts-as conditionals. In this way we are able to face the philosophical problem known as Jorgenson’s dilemma [10], which roughly says that a proper logic of norms is impossible because norms do not have truth values. Most formal systems reasoning about norms and time [11, 12, 13, 14, 15, 16, 17, 10, 18, 19] are restricted to obligations, prohibitions and permissions only, and do not consider the norms explicitly. The relation between our approach to normative systems and modal logics like deontic logic and conditional logic is not discussed in the present paper. Also it is beyond the scope of this paper to discuss the role of constitutive norms in creating or deleting other constitutive or regulative norms, for example to define contracts, as discussed in [20].

Despite the practical motivation, our formal approach also touches a long standing philosophical debate on the interplay between constitutive and regulative norms, because all earlier common frameworks to study this interplay do not consider the influence of time. Searle first claimed that constitutive rules are necessary to define institutions, and Ruiters [21] emphasizes the necessity and interplay of both kinds of rules. Ross [22] argues in the famous *tû-tû* example that intermediate concepts created by constitutive rules are superfluous, in the sense that the same deontic conclusions can be obtained through inferences directly connecting brute facts and deontic qualifications. Anderson [23] argues, instead, that regulative rules are not necessary and that they can be defined in terms of violations, and Grossi [5] argues that regulative rules can be defined in terms of constitutive rules. Finally, Boella and van der Torre [24] argue that regulative rules are necessary also in the definition of constitutive rules, using an example from procedural law.

The layout of this paper is as follows. In Section 2 we consider constitutive norms and institutional facts. In Section 3 we consider regulative norms and obligations that are preserved until they are obeyed. In Section 4 we consider violation labelings, and corresponding notions of violation redundancy and violation equivalence.

2 Institutional Equivalence of Constitutive Normative Systems

To formalize the example, we start with the definitions for ‘tree’ and ‘normative system’. The nodes of a tree represent moments in time, and we say that $\phi \in H(n)$ if brute fact ϕ holds at node n of the tree. For example, $esign_{jp} \in H(n)$ represents that there is an electronic signature of agent j at moment n . We assume that each node is complete, in the sense that each brute fact is either true or false. In the figure we represent the brute facts that hold in the circle, and all brute facts not represented are therefore

assumed not to hold. Moreover, we assume that we can use propositional connectives to talk about the brute facts. So, for example, we can say that at some moment in time “there is an electronic signature of agent j and there is a selling contract”. To keep the figure readable, we do not explicitly represent any of these logical consequences in the figure. These two assumptions imply that $H(n)$ is a maximally consistent set of formulas of a propositional logic. Maximal consistent sets satisfy exactly all the properties we need, like, if $\phi \vee \psi \in H(n)$ then $\phi \in H(n)$ or $\psi \in H(n)$, and closure under logical consequence.

Definition 1 (Tree). Let L_P be a propositional language built on a set of brute facts P , Let L_I be a propositional language built from a set of institutional facts I , and let L be a propositional language built from $P \cup I$. A tree is a tuple $T = \langle N, E, H \rangle$ where N is a set of nodes, $E \subseteq N \times N$ is a set of edges obeying the tree properties, and $H : N \rightarrow 2^{L_P}$ is a labeling function assigning to each node n a maximally consistent set of propositional formulas from L_P .

Constitutive norms like (*esign_{jp} counts-as sign_{jp} in contr*) are used to define institutional facts in terms of brute facts and other institutional facts.

Definition 2 (Constitutive norms). A constitutive norm “ x counts-as y in c ” is represented by a formula “fact” $x \in L$, a formula “institutional fact” $y \in I$ and a formula “context” $c \in L_P$, and written as (x counts-as y in c). A constitutive normative system is a set of norms $CN = \{(x_1 \text{ counts-as } y_1 \text{ in } c_1), \dots, (x_n \text{ counts-as } y_n \text{ in } c_n)\}$.

In our approach, constitutive norms are used to detach counts-as conditionals and institutional facts at each node of a tree. We call it the operational semantics for the norms, because the way we label the temporal structure determines the meaning of the constitutive norms. The counts-as conditionals are pairs of a propositional formula and an institutional fact, and the institutional facts are an institutional labeling of the temporal structure. We assume that the institutional facts at a node are again a maximal consistent set, representing the ideal alternative for the node, and this logical closure is again not visualized in Figure [□](#).

Definition 3 (Constitutive norm semantics). A counts-as labeling is a function $CA : N \rightarrow 2^{L \times I}$ and an institutional labeling is a function $F : N \rightarrow 2^I$. The constitutive norm semantics of a normative system CN is the unique counts-as and institutional labeling $CA : N \rightarrow 2^{L \times I}$ and $F : N \rightarrow 2^I$ such that for each node n , $CA(n)$ and $F(n)$ are the minimal sets such that:

1. for all norms (i, o, c) and all nodes n , if $c \in H(n)$, then $(i, o) \in CA(n)$.
2. if $(i, o) \in CA(n)$ and i is a propositional consequence of i' , then $(i', o) \in CA(n)$.
3. for all counts-as conditionals $(i, o) \in CA(n)$ and all nodes n , if i is a propositional consequence of $H(n) \cup F(n)$, then $o \in F(n)$.
4. if φ is a propositional consequence of $F(n)$ then $\varphi \in F(n)$.

The following example illustrates how the constitutive norms of the running example are used to label the branching time structure. We distinguish between so-called factual and institutional detachment. The former is based on a match between the condition of the norm and the facts, and the latter kind is based on a match between the condition and the institutional facts. For institutional facts, we want both kinds of detachment.

Example 1. The counts-as conditionals $(esign_{jp}, sign_{jp}) \in CA(n)$ and $(sign_{jp}, owe_{jp}) \in CA(n)$ are detached in all nodes n with $contr \in H(n)$, also where $esign_{jp}$ or $sign_{jp}$ does not hold. In the figure, we represent the counts-as conditional $(esign_{jp}, sign_{jp}) \in CA(n)$ by writing $CA(esign_{jp}, sign_{jp})$ next to node n , and so on. $CA(esign_{jp}, sign_{jp})$ without $esign_{jp}$ can be read as a counterfactual: if $esign_{jp}$ would have been true, then $F(sign_{jp})$ would have been the case. The institutional fact $F(sign_{jp})$ is detached in all $esign_{jp} \wedge contr$ nodes, and the institutional fact $F(owe_{jp})$ is detached in all $contr$ nodes where also $F(sign_{jp})$ is detached. None of the counts-as conditionals or institutional facts persists in time.

We now define how to reason about norms, institutions and time. We define equivalence of normative systems as equivalence of the labeling they give rise to, and a norm is redundant when it does not affect the labeling of the temporal structure. For many applications, we are interested only in the institutional facts, not in the counts-as conditionals which can be derived from a constitutive normative system. This leads to two notions of redundancy.

Definition 4 (Equivalence and redundancy). *Two constitutive normative systems CN_1 and CN_2 are counts-as (institutionally) equivalent if and only if for each temporal structure T , the counts-as (institutional) labeling by CN_1 is identical to the counts-as (institutional) labeling by CN_2 . A norm $(i, o, c) \in CN$ is constitutively (institutionally) redundant in constitutive normative system CN if and only if CN and $CN \setminus \{(i, o, c)\}$ are counts-as (institutionally) equivalent.*

The following result characterizes *all* properties which hold for constitutive normative systems. We do not detail the proofs of this and following theorems.

Theorem 1 (Redundant constitutive norms). *In a constitutive normative system CN , a constitutive norm $(x \text{ counts-as } y \text{ in } c) \in CN$ is counts-as redundant if we can derive it from $CN \setminus \{(x \text{ counts-as } y \text{ in } c)\}$ using replacement of logical equivalents in input and context, together with the following rules:*

$$\frac{(x_1 \text{ counts-as } y \text{ in } c_1)}{(x_1 \wedge x_2 \text{ counts-as } y \text{ in } c_1 \wedge c_2)} SIC \quad \frac{(x \text{ counts-as } y \text{ in } c_1)(x \text{ counts-as } y \text{ in } c_2)}{(x \text{ counts-as } y \text{ in } c_1 \vee c_2)} ORC$$

$(x \text{ counts-as } y \text{ in } c) \in CN$ is institutionally redundant when we can derive it from $CN \setminus \{(x \text{ counts-as } y \text{ in } c)\}$ with the above rules, together with the following ones:

$$\frac{(x_1 \text{ counts-as } y \text{ in } c)(x_2 \text{ counts-as } y \in c)}{(x_1 \vee x_2 \text{ counts-as } y \text{ in } c)} ORX$$

$$\frac{(x_1 \wedge x_2 \text{ counts-as } y \text{ in } c)}{(x_1 \text{ counts-as } y \text{ in } c \wedge i_2)} I2C \quad \frac{(x \text{ counts-as } y \text{ in } c_1 \wedge c_2)}{(x \wedge c_2, y, c_1)} C2Y$$

$$\frac{(x \text{ counts-as } y_1 \text{ in } c), (x \wedge y_1 \text{ counts-as } y_2 \text{ in } c)}{(x \text{ counts-as } y_2 \text{ in } c)} CT \quad \frac{(x \text{ counts-as } y_1 \wedge y_2 \text{ in } d)}{(x \text{ counts-as } y_1 \text{ in } d)} WY$$

Reasoning about normative systems is illustrated in the running example.

Example 2 (Continued from Example 1). Let

$$CN = \{(esign_{jp} \text{ counts-as } sign_{jp} \text{ in } contr), (sign_{jp} \text{ counts-as } owe_{jp} \text{ in } contr)\}$$

The norm $(esign_{jp} \text{ counts-as } owe_{jp} \text{ in } contr)$ is institutionally redundant in normative system

$$CN' = CN \cup \{(esign_{jp} \text{ counts-as } owe_{jp} \text{ in } contr)\}$$

due to institutional detachment represented by inference rule CT. However, it is not counts-as redundant, since the counts-as rule $(esign_{jp} \text{ counts-as } owe_{jp})$ cannot be detached using the two rules of our running example.

Proposition 1 (Monotony of labeling functions). *A labeling function $Lab : N \rightarrow S$, with N a set of nodes in a temporal structure T , and S either a set of counts-as pairs, or any other set of labels build from propositional formulas, is monotonic if and only if for each node n , the labels $Lab(n)$ in normative system S are a subset of the labels $Lab'(n)$ in any normative system S' with $S \subseteq S'$. The counts-as and institutional labeling are monotonic.*

The following proposition follows from monotony of the counts-as labeling function.

Proposition 2. *Two constitutive normative systems CN_1 and CN_2 are counts-as (institutionally) equivalent if and only if each norm of CN_1 is counts-as (institutionally) redundant when added to CN_2 , and vice versa.*

Example 3 (Continued from Example 2). The two normative systems

$$CN = \{(esign_{jp} \text{ counts-as } sign_{jp} \text{ in } contr), (sign_{jp} \text{ counts-as } owe_{jp} \text{ in } contr)\}$$

and

$$CN' = CN \cup \{(esign_{jp} \text{ counts-as } owe_{jp} \text{ in } contr)\}$$

are institutionally equivalent.

3 Obligation Equivalence of Regulative Norms with Deadlines

It has been argued [25] that for temporal regulative norms, deadlines are of essential importance. Therefore, for regulative normative systems we extend the norms with deadlines, and generalize the persistency semantics introduced in [8] in order to deal with these deadlines. Extensions with, for example, explicit sanctions, as well as other kinds of norms like permissive and procedural norms, are left for further research. Following conventions in input/output logic [26, 27], in [8] we write a conditional norm “if i , then o is obligatory” as a pair of propositional formulas (i, o) . Similarly, in this paper, we write a conditional norm “if i , then o is obligatory before deadline d ” as a triple of propositional formulas (i, o, d) .

Consequently, since in this paper we focus on temporal reasoning with norms, we do not consider contrary-to-duty norms stating what should hold in sub-ideal nodes [28, 29] or dilemmas stating various optimal alternatives for a node [30]. For a discussion on the various problems with such norms and approaches such as priorities to deal with these problems, see [31].

Example 4 (Continued). Consider again the temporal structure in Figure 1. From the root node visualized on the left side of the figure we can access only one node in which “John owes Peter \$1000”, $F(owe_{jp})$. From this node, we can reach two other nodes, one in which “John pays Peter \$1000”, pay_{jp} , and one in which it does not happen. The norms are “if owe_{xy} , then obligatory pay_{xy} before d ” (owe_{xy}, pay_{xy}, d), and “if pay_{xy} , then obligatory $receipt_{yx}$ before e ” ($pay_{xy}, receipt_{yx}, e$). Here x and y are variables ranging over the set of agents, in the sense that each norm is treated as a set of propositions based norms, for each instance of the agent variables.

As discussed in [8, 17], we do not want to have iterated or deontic detachment, in the sense that Peter is not obliged to give John a receipt until he has given him the money. Moreover, in contrast to the constitutive norm semantics, if John pays Peter, then Peter has to give John the receipt directly or *at some point in the future*.

Example 5 (Continued). Consider the desired labeling of the temporal structure in Figure 1. In the circumstance that John owes Peter \$1000 but he has not paid him yet, Peter does not have the obligation to write a receipt (see bottom path). That obligation arises only when John fulfils his obligation by paying Peter (top path). Consequently, we have factual detachment without iterated or deontic detachment. Moreover, for each norm, we have that if the condition is true in a node, then the obligation holds for the node itself. Moreover, if the obligation is not fulfilled, then it also holds for all successor nodes (see, e.g., bottom path).

In addition to factual detachment and persistence, we assume that the obligatory formulas at a node are a deductively closed set. This logical closure is not visualized in Figure 1 and it implies that we do not consider contrary-to-duty norms.

Definition 5 (Persistent norm semantics). *The persistent norm semantics of a normative system S is the unique obligation labeling $O : N \rightarrow 2^L$ such that for each node n , $O(n)$ is the minimal set such that:*

1. *for all norms (i, o, d) , nodes n_1 , and paths (n_1, n_2, \dots, n_m) with $m \geq 1$, if i is a consequence of $H(n_1) \cup F(n_1)$ and $o \vee d$ is not a consequence of $H(n_k) \cup F(n_k)$ for $1 \leq k \leq m - 1$, then $o \in O(n_m)$.*
2. *if φ is a propositional consequence of $O(n)$ then $\varphi \in O(n)$.*

We might have used more complicated labels. For example, in [9] we use dyadic obligations to model obligations which can be fulfilled before the antecedent has become true. In our running example, Peter could give the receipt to John before he receives the money (not illustrated in the figure). Moreover, we could also label the temporal structure with deadline obligations like $O(p \leq d)$, as studied in [25]. However, that would be less in line with the central paradigm in this paper, where the semantics of norms is defined in terms of the obligations they give rise to. Deadlines in the norms then are interpreted as intervals in the temporal models in which the corresponding obligation holds. It would thus be superfluous to label the temporal structures with deadline obligations. So, although monadic modal logic has been considered too simple for formalizing deontic reasoning in case of implicit normative systems, here monadic modal logic seems sufficient.

Redundancy and equivalence for persistent obligations is analogous to reasoning about constitutive norms in Definition 4.

Definition 6 (Obligation equivalence and redundancy). *Two regulative normative systems RN_1 and RN_2 are obligation equivalent if and only if for each temporal structure T , the obligation labeling by RN_1 is identical to the obligation labeling by RN_2 . In regulative normative system RN , a norm $(i, o, d) \in RN$ is obligation redundant if and only if RN is obligation equivalent to $RN \setminus \{(i, o, d)\}$.*

Without proof, we give the following result.

Theorem 2 (Redundant regulative norms). *In a regulative normative system RN , a regulative norm $(i, o, d) \in RN$ is obligation redundant under the persistence semantics when we can derive it from $RN \setminus \{(i, o, d)\}$ using replacement of logical equivalents in input, output and deadline, together with the following rules:*

$$\begin{array}{c} \frac{}{(\perp, \perp, \top)} \perp \quad \frac{}{(\top, \top, \top)} \top \quad \frac{(i_1, o, d)}{(i_1 \wedge i_2, o, d)} SI \quad \frac{(i, o_1 \wedge o_2, d)}{(i, o_1, d)} WO \\ \\ \frac{(i_1, o, d)(i_2, o, d)}{(i_1 \vee i_2, o, d)} ORI \quad \frac{(i, o, d_1 \wedge d_2)}{(i, o, d_1)} WD \quad \frac{(i, o, d)}{(i, o, d \wedge o)} OSD \\ \\ \frac{(i, o_1, \top), (i, o_2, \top)}{(i, o_1 \wedge o_2, i)} AND_{\top} \quad \frac{(i, o, p)(p, o, q)}{(i, o, q)} TRD \end{array}$$

We do not have redundancy of transitivity because we no longer have deontic detachment. The conjunction rule AND holds only for obligations that do not persist in time. It does not even hold for obligations sharing the same deadline, as illustrated by Example 6.

Example 6 (Continued). Consider the regulative normative systems

$$RN_1 = \{(owe_{xy}, pay_{xy} \wedge receipt_{yx}, d)\}$$

$$RN_2 = \{(owe_{xy}, pay_{xy}, d), (owe_{xy}, receipt_{yx}, d)\}$$

$$RN_3 = \{(owe_{xy}, pay_{xy}, d), (owe_{xy}, pay_{xy} \rightarrow receipt_{yx}, d)\}$$

If $n \models owe_{jp} \wedge pay_{jp} \wedge \neg receipt_{pj}$, then we have $O(n)$ is the consequence set of $pay_{jp} \wedge receipt_{pj}$. Hence, at node n we cannot discriminate among the three normative systems. However, we have for any successor node of n that using RN_1 we have $pay_{jp} \wedge receipt_{pj} \in O(m)$, because the obligation for $pay_{jp} \wedge receipt_{pj}$ persists, using RN_2 we have only $receipt_{pj} \in O(m)$, because we have that only the obligation for $receipt_{pj}$ persists, and using RN_3 , we have only $pay_{jp} \rightarrow receipt_{pj} \in O(m)$, because only the obligation for $pay_{jp} \rightarrow receipt_{pj}$ persists. The example also illustrates that the three normative systems RN_1 , RN_2 and RN_3 are *not* equivalent.

4 Violation Equivalence of Regulative Norms with Deadlines

Deadlines are used with persistent norms to determine whether the norm is violated. For instance, in Figure 1 the obligation $O(\text{pay}_{jp})$ persists until d , without visiting a state where pay_{jp} , which results in a violation. Below we define violation equivalence for regulative norms. First we define a violation labeling of temporal structures.

Definition 7 (Violation semantics). A violation labeling is a set $V \subseteq N$ such that $n_m \in V$ if and only if there is a norm (i, o, d) , a node n_1 and a path (n_1, n_2, \dots, n_m) with $m \geq 1$, such that i is a consequence of $H(n_1) \cup F(n_1)$ and $o \vee d$ is not a consequence of $H(n_k) \cup F(n_k)$ for $1 \leq k \leq m - 1$, and $\neg o \wedge d$ is a consequence of $H(n_m) \cup F(n_m)$.

Violation equivalence and redundancy for persistent obligations are defined in the same way as for regulative norms in Definition 6. Even if we are interested in violations only, and thus not in the obligation, we can distinguish the following two notions of violation equivalence. Local violation equivalence distinguishes between violations at different nodes, while global violation equivalence identifies any two violations on a single path.

Definition 8 (Violation equivalence). Two regulative normative systems RN_1 and RN_2 are locally violation equivalent if and only if for each temporal structure T , the violation labeling by RN_1 is identical to the violation labeling by RN_2 . Two regulative normative systems RN_1 and RN_2 are globally violation equivalent if and only if for each temporal structure T , for each path σ in T , there is an $n \in \sigma$ such that $n \in V$ for RN_1 if and only if there is $n \in \sigma$ such that $n \in V$ for RN_2 .

Definition 9 (Violation redundancy). In regulative normative system RN , a norm $(i, o, d) \in RN$ is (locally / globally) violation redundant if and only if RN is (locally / globally) violation equivalent to $RN \setminus \{(i, o, d)\}$.

Theorem 2 says, among other things, that deadlines can be weakened. That is, if we weaken the deadline of a regulative norm from $d_1 \wedge d_2$ to d_1 , we get a redundant norm (see 12 for similar properties in a modal approach to deontic deadlines). However, Theorem 3 shows that if we identify a norm with the nodes in which it is violated, which seems a reasonable alternative way of viewing norms, then this property does no longer hold. In general, weaker deadlines may be satisfied sooner, which means that violations also can occur sooner. So there is a difference between obligation equivalence of regulative normative systems and violation equivalence.

Theorem 3 (Violation redundancy regulative norms). In a regulative normative system RN , a regulative norm $(i, o, d) \in RN$ is locally violation redundant if we can derive it from $RN \setminus \{(i, o, d)\}$ using replacement of logical equivalents in input, output and deadline, together with the following rules:

$$\begin{array}{c} \frac{}{(\perp, \perp, \top)} \perp \quad \frac{}{(i, o, d \wedge o)} OD \quad \frac{(i_1, o, d)}{(i_1 \wedge i_2, o, d)} SI \quad \frac{(i, o_1 \wedge o_2, d)}{(i, o_1, d)} WO \\ \frac{(i_1 \wedge i_2 \wedge i_3, o, i_1 \wedge i_2)}{(i_1 \wedge i_2 \wedge i_3, o, i_1)} RWD \quad \frac{(i_1 \wedge i_2 \wedge i_3, o, i_1)}{(i_1 \wedge i_2 \wedge i_3, o, i_1 \wedge i_2)} RSD \end{array}$$

$$\frac{(i_1, o, d)(i_2, o, d)}{(i_1 \vee i_2, o, d)} OR \quad \frac{(i, o_1, \top)(i, o_2, \top)}{(i, o_1 \wedge o_2, i)} AND_{\top} \quad \frac{(i, o, d_1)(i, o, d_2)}{(i, o, d_1 \vee d_2)} ORD$$

For global violation redundancy we get, in addition:

$$\frac{(i, \neg i, i)}{(i, o, d)} V$$

5 Summary

Reasoning about norms and time is of central concern to the regulation of multiagent system behavior [2, 3]. In particular, for the study of social norms emerging in societies which aim at enforcing desirable group behavior, for the design of legal norms to meet institutional goals in electronic institutions, for the design of organizational norms to structure organizations and regulate agents playing a role in the organization, and for the study of norms in contracting in electronic commerce. Broersen and van der Torre [8, 9] show that the distinction between norms and obligations leads to a simple and therefore practical way to reason about norms, obligations of agents and time, and they illustrate the approach by discussing three ways to relate norms and obligations over time. Also they show how these three can be characterized, generalizing the non-temporal input/output logic framework. The approach of Broersen and van der Torre [8, 9] to reasoning about norms, obligations, time and agents takes three steps.

1. They assume a branching temporal structure representing how propositions change over time, where the branches represent either uncertainty or alternative actions of the agents. Such a temporal structure can be generated using a formal language, like, for example, in model checkers such as Mocha [32] or action logics such as the causal calculator [33].
2. They use an algorithm that, given the input of the branching temporal structure and a set of norms, produces as output a labeling of the temporal structure with obligations. The algorithm determines the meaning of the norms, and it may therefore be considered an operational semantics of the norms. They give formal definitions for the possible labelings, enabling them to say when norms are equivalent or redundant.
3. Two normative systems are equivalent, when they lead to the same labeling. A norm is redundant in a normative system, if removing the norm from the normative system leads to an equivalent normative system.

Broersen and van der Torre [8, 9] extend the input/output logic framework to reason about regulative norms and obligations in time. In the present paper we consider constitutive norms in this temporal framework. Just like regulative norms are used to detach obligations, constitutive norms are used to detach counts-as conditionals and institutional facts. Whereas Broersen and van der Torre [8, 9] show that the approach leads to a simple way to reason about norms and obligations in time, this paper shows that the new approach leads to a detailed study of various kinds of equivalence of normative systems.

In particular, we obtain the following results. Given a set of constitutive norms, we label temporal structures with counts-as conditionals and institutional facts, by using both factual and institutional detachment. We distinguish between counts-as and institutional equivalence of two constitutive normative systems, and we show that the latter leads to many logical properties. We show that a simple extension of the persistence condition leads to the extension of the labeling of temporal structures with persistent regulative norms with deadlines. Instead of persisting until the obligation is fulfilled, an obligation persists until the obligation is fulfilled or the deadline is reached. The obligations themselves do not refer to the deadline explicitly. Moreover, we label temporal structures with violations by checking whether the obligations are fulfilled at the deadline. We distinguish local and global violation redundancy and equivalence, depending on whether the moment of violation is taken into account, and show that the logics of obligation redundancy and the two kinds of violation redundancy are subtly different. For example, with obligation redundancy we have weakening of the deadline, which does not hold for violation redundancy.

References

1. Arcos, J., Esteva, M., Noriega, P., Rodríguez, J., Sierra, C.: Engineering open environments with electronic institutions. *Journal on Engineering Applications of Artificial Intelligence* 18(2), 191–204 (2005)
2. Boella, G., van der Torre, L., Verhagen, H. (eds.): *Computational and Mathematical Organization Theory. NorMAS 2005*, vol. 12(2-3) (2006)
3. Boella, G., van der Torre, L., Verhagen, H. (eds.) *Normative Multi-agent systems*. In: *Procs. of NorMAS 2007, Dagstuhl Seminar proceedings 07122* (2007)
4. Gaudou, B., Longin, D., Lorini, E., Tummolini, L.: Anchoring institutions in agents' attitudes: Towards a logical framework for autonomous multi-agent systems. In: *Procs. of AAMAS 2008* (2008)
5. Grossi, D.: Pushing Anderson's envelope: The modal logic of ascription. In: van der Meyden, R., van der Torre, L. (eds.) *DEON 2008. LNCS*, vol. 5076, pp. 263–277. Springer, Heidelberg (2008)
6. Jones, A., Sergot, M.: A formal characterisation of institutionalised power. *Journal of IGPL* 3, 427–443 (1996)
7. Searle, J.: *Speech Acts: an Essay in the Philosophy of Language*. Cambridge University Press, Cambridge (1969)
8. Broersen, J., van der Torre, L.: Reasoning about norms, obligations, time and agents. In: *Procs. of PRIMA 2007. LNCS*. Springer, Heidelberg (to appear)
9. Broersen, J., van der Torre, L.: Conditional norms and dyadic obligations in time. In: *Procs. of 18th European Conference on Artificial Intelligence (ECAI 2008)* (2008)
10. Makinson, D.: On a fundamental problem of deontic logic. In: McNamara, P., Prakken, H. (eds.) *Norms, Logics and Information Systems. New Studies on Deontic Logic and Computer Science*, pp. 29–54. IOS (1999)
11. Alchourrón, C., Bulygin, E.: *Normative Systems*. Springer, Wien (1971)
12. Broersen, J.: Strategic deontic temporal logic as a reduction to ATL, with an application to chisholm's scenario. In: Goble, L., Meyer, J.-J.C. (eds.) *DEON 2006. LNCS*, vol. 4048, pp. 53–68. Springer, Heidelberg (2006)
13. Broersen, J., Brunel, J.: What I fail to do today, I have to do tomorrow: a logical study of the propagation of obligations. In: *Procs. of CLIMA VIII. LNCS*, vol. 5056, pp. 82–99. Springer, Heidelberg (2008)

14. Governatori, G., Rotolo, A., Sartor, G.: Temporalised normative positions in defeasible logic. In: ICAIL 2005: Proceedings of the 10th international conference on Artificial intelligence and law, pp. 25–34. ACM, New York (2005)
15. Horty, J.: *Agency and Deontic Logic*. Oxford University Press, Oxford (2001)
16. Loewer, B.: Dyadic deontic detachment. *Synthese* 54, 295–318 (1983)
17. Makinson, D.: Five faces of minimality. *Studia Logica* 52, 339–379 (1993)
18. Meyer, J.J.C.: A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic* 29(1), 109–136 (1988)
19. van Eck, J.: A system of temporally relative modal and deontic predicate logic and its philosophical applications. *Logique et Analyse* 25, 339–381 (1982)
20. Boella, G., van der Torre, L.: A game theoretic approach to contracts in multiagent systems. *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews* 36(1), 68–79 (2006)
21. Ruiters, D.: A basic classification of legal institutions. *Ratio Juris* 10(4), 357–371 (1997)
22. Ross, A.: Tû-tû. *Harvard Law Review* 70(5), 812–825 (1957)
23. Anderson, A.: A reduction of deontic logic to alethic modal logic. *Mind* 67, 100–103 (1958)
24. Boella, G., van der Torre, L.: Substantive and procedural norms in normative multiagent systems. *Journal of Applied Logic* 6(2), 152–171 (2008)
25. Broersen, J.: On the logic of being motivated to achieve ρ , before δ . In: Alferes, J.J., Leite, J. (eds.) *JELIA 2004*. LNCS, vol. 3229, pp. 334–346. Springer, Heidelberg (2004)
26. Makinson, D., van der Torre, L.: Input-output logics. *Journal of Philosophical Logic* 29(4), 383–408 (2000)
27. Makinson, D., van der Torre, L.: Constraints for input-output logics. *Journal of Philosophical Logic* 30(2), 155–185 (2001)
28. Chisholm, R.: Contrary-to-duty imperatives and deontic logic. *Analyse* 24, 33–36 (1963)
29. Forrester, J.: Gentle murder, or the adverbial samaritan. *The Journal of Philosophy* 81, 193–197 (1984)
30. van Fraassen, B.: Values and the hearts command. *The Journal of Philosophy* (1973)
31. Hansen, J., Pigozzi, G., van der Torre, L.: Ten philosophical problems in deontic logic. In: *Normative Multi-agent systems, Procs. of NorMAS 2007*. Dagstuhl Seminar proceedings 07122 (2007)
32. Alur, R., Henzinger, T., Mang, F.Y.C., Qadeer, S., Rajamani, S., Tasiran, S.: Mocha: Modularity in model checking. In: Y. Vardi, M. (ed.) *CAV 1998*. LNCS, vol. 1427, pp. 521–525. Springer, Heidelberg (1998)
33. Artikis, A., Sergot, M., Pitt, J.: Specifying electronic societies with the causal calculator. In: Giunchiglia, F., Odell, J.J., Weiss, G. (eds.) *AOSE 2002*. LNCS, vol. 2585, pp. 1–15. Springer, Heidelberg (2003)

When to Use a Multi-Agent System?

Paul Bogg¹, Ghassan Beydoun², and Graham Low¹

¹ University of New South Wales
{paulb, g.low}@unsw.edu.au

² University of Wollongong
beydoun@uow.edu.au

Abstract. The decision of which development approach to adopt (e.g. traditional, object oriented, agent oriented) is often taken after identifying the specific features of the problem. If *agent oriented software engineering* (AOSE) is to be taken seriously as a software engineering paradigm, then a clearly identifiable set of criteria of when to apply it, as distinguished from other alternatives such as object-oriented practices, is necessary. The paper is part of an effort to determine when is best to adopt a Multi Agent System approach, identifying a number of critical factors to include in a decision framework.

1 Introduction

Agents are highly autonomous, situated and interactive software components. They autonomously sense their environment and respond accordingly. A Multi Agent System (MAS) is a collection of interacting agents which are highly autonomous, situated and interactive software components. They autonomously sense their environment and respond accordingly. Coordination and cooperation between agents that possess diverse knowledge and capabilities facilitate the achievement of global goals that cannot be otherwise achieved by a single agent working in isolation [1]. MASs have been shown to be highly appropriate for the engineering of open, distributed or heterogeneous systems [2-4]. While many AOSE methodologies exist with each focusing on a specific class of problems (e.g. MaSE [5], GAIA [6], PROMETHEUS [7] and TROPOS [8]), a clear and established set of criteria for determining if, and when, an agent approach should be used remains absent. This no doubt has contributed to the delay in the much anticipated adoption of MAS as a preferred development approach in many medium to large-scale projects.

This paper is a part of our ongoing research to remove barriers to the successful adoption of AOSE. We present a preliminary multi-staged approach to identifying a set of criteria to be included in a decision framework that will assist in determining if, and to what degree, a MAS is suitable to solving a particular problem.

2 Related Work

Deciding if a MAS approach should be used to solve a particular problem requires identifying its *suitability* and estimating its *relative cost* (versus alternative approaches)

in terms of money, expertise and time. An answer to the question “*Should one use a MAS approach to solve this particular problem?*” is based on an acceptable degree of suitability combined with an acceptable level of cost. Not to understate the importance of a cost assessment, the primary focus in this paper is in defining a measure of *suitability* defined as the extent to which the properties of a MAS are appropriate to the successful design and implementation of the application system.

MASs have originally been proposed as a new software-based solution providing advantages over existing alternatives for a *particular family of problems* [1]. In Wooldridge [1], the following four criteria have been identified to establish the suitability of a MAS:

- When the environment is open or at least highly dynamic, uncertain or complex;
- When agents are a natural metaphor;
- When data, control or expertise are distributed; and/or
- When extending legacy systems.

These criteria provide some guidance but are open to interpretation particularly for problems not previously addressed by a MAS solution. For instance, it is left to the designer to decide when agents are a natural metaphor. Two other similar sets of criteria have been proposed by De Wolf and Holvoet [9] and EURESCOM [10] respectively. The De Wolf and Holvoet’s [9] criteria are:

- When there is a need to maintain overall system properties (e.g. self-optimising, self-healing, self-configuring);
- When there is decentralised or distributed information availability (e.g. in competitive situations, or communication failure somewhere); and/or
- When there is high dynamics and frequent changes (e.g. robustness is required, adaptability to environment changes).

EURESCOM [10]’s criteria are:

- When complex/diverse types of communication are required;
- When the system must perform well in situations where it is not practical/ possible to specify its behaviour on a case-by-case basis;
- When negotiation, cooperation, and competition between entities is involved;
- When the system must act autonomously; and/or
- When high modularity is required (for when the system is expected to change).

Both preceding sets of criteria further help to identify when an agent-based solution may be appropriate, but they are still quite general. They do not offer clear guidelines as to when MASs may be the *preferable* approach. For instance the criteria “*when there is decentralised or distributed information availability*” can actually be addressed by non agent-oriented system implementations. By taking account of the existing wide range of MAS applications and applying a methodical analysis of their common features, we provide two detailed sets of features detailing features of problems suited for MAS and features of solutions that can be expected from a MAS solution. This raises further questions: to what extent is a MAS suited to address the features of a particular problem? Are some features more important in determining this suitability? And finally, how can one validate the appropriateness of any defined

set of features? To provide answers for these questions we construct a framework that can be applied to a problem instance to assess the suitability of a MAS solution and discuss ongoing work.

3 MAS Features for Determining Suitability

In this section, we lay the foundations to construct a framework to assess whether a MAS is an appropriate solution for a given problem. We first overview properties of agents and MAS. These properties we know that any MAS can be designed to have. After overviews of 25 existing MAS applications from the literature, we establish a comprehensive set of features that affect the decision of whether or not to adopt an agent-oriented approach and we relate these features to the known properties of MAS and agents.

Individual agents show varying levels of autonomy, social ability, reactivity and pro-activity and/or reasoning capabilities. *Autonomy* is the independent control that an agent has over tasks performed as well as its control over the acquisition of inputs and resources required for its tasks. *Reasoning capabilities* is the ability that the agent has to identify reasons for beliefs, conclusions or performing tasks. *Social ability* is the ability that an agent has to interact with other agents or other components of the environment. A MAS solution has *system-level* properties that result from the interactions between the individual agents in the system. *Self-organisation* as a key system-level property of MASs which is a process allowing the system to change its internal organisation to adapt to changes in its goals and the environment without explicit external control [11]. Various characteristics of self-organising systems are (adapted from [11]): *decentralised control*, *self-maintenance*, *adaptivity*, and *convergence*. *Decentralised control* arises in the absence of a central authority when the system operation is governed by the interactions between individual agents. *Self-maintenance* is where a system has the capacity to reproduce or repair itself. *Adaptivity* is the capability of the system to reorganise itself when the environment or goals change in order to continue operating. *Convergence* is the capability of the system to reach a stable state. For example, if the interactions between individual air conditioning unit agents keep the building temperature constant, then we might say the system has decentralised control, is adaptive to changes in the environment, and converges to a stable state.

To establish and validate a comprehensive set of features that affect the decision of whether or not to adopt an agent-oriented approach, we follow an iterative approach based on an initial analysis of specific instances of problems for which MASs have been implemented. These instances are distinct and are identified using a combination of literature and preliminary discussions with experts in MAS development. In all, 25 different MAS solutions from the literature were analysed (Table 1 provides a sampling). A set of features that characterise this spectrum of problems is identified. In deriving this set, every effort was made to make it domain independent and applicable to new domains not previously considered as candidates for agent-based solutions. Features are identified into two categories: problem-related features and solution-related features (degree of distribution, efficiency, etc).

To elaborate and illustrate the resultant features, we use the following two examples which have been implemented using a MAS solution:

Example A. *An Intrusion Detection System (IDS) adapted from [12]: IDS has a function by which mobile agents trace intruders, collect information related to the intrusion along the intrusion-route, and decide whether, in fact, an intrusion has occurred. Intrusions are mainly divided into two types: break-ins from outside the local area network (LAN) and those from inside the LAN. Intruders tend to attack the less-protected hosts first, gradually approaching hosts armed with stronger protection, ultimately working up to and reaching their target hosts. Commonly, administrators do not notice the intrusion. Furthermore, the administrators cannot trace the origin of an intrusion after the network connection has closed even if the intrusion has been detected. Types of attack include data driven attacks on applications, host-based attacks such as privilege escalation, and malware (viruses, trojans).*

Example B. *A Battlefield Information System adapted from [13]: A battlefield simulation system provides the commander with tactical and strategic intelligence during a conflict situation. To accomplish this goal, various types of sensors are used to detect events and objects of interest. This sensor data can then be combined, or fused, with other sensor data to provide a commander with a much clearer, more complete picture of the battlefield. Due to the nature of war, there is also a high probability that a percentage of these sensors will become disabled during the battle. However, when these sensors are lost or destroyed, the information produced by those sensors must still be provided to the battlefield commander.*

3.1 Problem-Related Features

Any software solution operates within an environment. An environment is a computer system, network or existing software within which the software solution is embedded. In order to perform tasks, the software solution may require input from the environment. The environment properties are themselves features related to the problem, we identify the following:

- Dynamic Environment – input from the environment periodically changes;
- Uncertain Environment – input from the environment is possibly untrue;
- Complex Environment – input from the environment is difficult to understand and/or act upon. It may involve multiple interrelated decisions; and
- Open Environment – new components of the system, network, or existing software can appear during runtime.

In Example A, the input from the environment is *dynamic*. Part of the difficulty in designing an IDS solution is that the solution is required to handle repeated and new forms of attack. The extent to which the environment is dynamic depends on the specific instance to which the IDS is applied. The input from the environment may also be *open*. For networked systems, an IDS may be required to monitor attacks on new computers or network additions that may occur during runtime.

In Example B, the input from the environment is *dynamic*. A feature of the battlefield simulation system is to handle new events and objects of interest. In this example, the input is *open* because of the likelihood that existing sensors may be destroyed and/or replaced.

Software solutions often need to *interact* with other software, or components within their environment. Interactions might be as simple as an enquiry for information, or as complex as a negotiation. The properties of these interactions are themselves features of the problem (as adapted from [14]):

- Negotiation – process of coming to agreement on a set of conflicting issues; and
- Deliberation – interactions to establish cooperation on tasks.

For example, software automating an economic business process may require negotiation with external business partners to acquire goods and services.

In Example B, when sensors are destroyed, some *deliberation* between the remaining sensors is required in order to compensate.

3.2 Solution-Related Features

Software solutions may need to perform tasks in or acquire input from geographically separate locations. This distributedness of the solution has the following facets which are also features of the solution:

- Distributed Data – input from the environment is from geographically separate locations;
- Distributed Resources – resources required to perform a task are in geographically separate locations; and
- Distributed Tasks – tasks are performed in geographically separate locations.

In Example A, if the environment is a computer network, then the input is provided by sensors around the network – so there is distributed data. Since the tasks are required to be performed remotely there is a need for *distributed tasks*. (The feature of task distribution might be dependent on the framing of the problem. A similar IDS problem might not require distributed tasks.)

In Example B, *distributed data* is acquired by the battlefield simulation system. While the sensors used to acquire the data might be distributed, a significant proportion of tasks would be performed centrally.

In addition to the above, we identify a set of non-functional requirements that are themselves features of a MAS solution:

- Efficiency – in performing tasks, acquiring inputs, and resource expenditure;
- Reliability – in performing tasks (consistency in doing what it's supposed to do);
- Robustness – in performing tasks (continues to perform in the face of adversity);
- Flexibility – in performing different/new tasks;
- Responsiveness – in performing tasks in response to inputs at runtime;
- Indeterminism at design time – in identifying which tasks to perform; and/or
- Concurrency – in performing tasks (at the same time).

In Example A, the IDS requires that the system be *reliable* in continuing to detect known and new forms of attack. *Robustness* is required in situations where network components fail. *Flexibility* is required in detecting new forms of attack. *Responsiveness* is imperative in order to promptly alert the system administrators of possible intrusions. *Concurrency* is required when many parts of the network are monitored

simultaneously. There is an expected compromise in general efficiency; however, new forms of attack are required to be detected *efficiently*.

In Example B, the system should be *reliable* so that it can continue to monitor the battlefield. *Robustness* is imperative since sensors can fail. *Responsiveness* is imperative for providing information. Flexibility, handling indeterminism, and high efficiency are not necessarily features of the system.

3.3 Relating MAS Designed Properties to Identified Features

As a first in constructing our framework to identify when a MAS is to be used, we highlight the relations between agent-level and system-level properties (from the solution perspective) to the features identified previously in the following way:

- Environment – Dynamic – Agents required to perform tasks in dynamic environments are *reactive* in response to changing inputs and resources, and have the ability to *reason* about these changes. As *autonomous* entities, agents have control over what to do as a result of undesirable or unexpected changes in order to achieve their goals. MASs have *adaptive* behaviour that adjusts to environment changes. For example, the battlefield simulation system adapts to the disabling of sensors in order to continue operation.
- Environment – Uncertain – Agents required to perform tasks in an uncertain environment can *reason* about the uncertainty of the inputs and resources required. As *autonomous* entities, agents have control over what to do as a result of uncertain inputs or resources in order to achieve their goals. As *social* entities, agents may exchange information to reduce uncertainty.
- Environment – Open – Agents required to perform tasks in an open environment can *reason* about new software components and the changes to the inputs and resources that they bring. As *social* entities, agents may communicate with new software components, or communicate about new components with other agents. MASs have *adaptive* behaviour that adjusts to new software components in order to preserve system operation. For example, an IDS adapts itself to monitor new computers on the network to maintain system-level security.
- Distributed – Data – Agents may acquire geographically distributed input in a *pro-active* or *reactive* way. As *social* entities, agents may exchange data with other geographically distributed agents. MASs have *decentralised control* over the acquisition or exchange of geographically distributed data. For example, the battlefield simulation system uses sensors distributed over a wide geographic area to exchange data when one or more sensors are disabled.
- Distributed – Resources – Agents may acquire geographically distributed resources in a *pro-active* or *reactive* way. As *social* entities, agents may exchange resources with other distributed agents. MASs have *decentralised control* over the acquisition or exchange of geographically distributed resources.
- Distributed – Tasks – Agents may be *pro-active* or *reactive* in performing tasks in remote locations. As *social* entities, agents may ask other agents to perform tasks remotely. MASs have *decentralised control* over the execution of tasks, meaning other agents in remote locations may be asked to perform tasks. For example, the IDS uses mobile agents to perform remote security-based tasks in a distributed network.

- Interactions – Negotiation – Agents may be *pro-active* or *reactive* in the process of negotiation in order to achieve their goals. As a *social* entity an agent may negotiate with one or more agents. As an *autonomous* entity, an agent may have control over what information is relevant to the negotiation. Where negotiation is needed to regulate the amount of resources amongst entities, MASs may have *convergent* behaviour that regulates how these resources are distributed. An example of this is in the stabilising of a global market price amongst competing agents in a market place [15].
- Interactions – Deliberation – Agents may be *pro-active* or *reactive* in cooperating with other agents in order to achieve their goals. As a *social* entity, an agent may communicate with other agents. As an *autonomous* entity an agent may have control over if, when and how to cooperate and what information is relevant to the cooperation. When cooperation is necessary to achieve a stable state, MASs may have *convergent* behaviour that regulates the continuity of the state. For example, air conditioning units may be required to cooperate in order to regulate global temperature, and a MAS solution provides the regulation of the global temperature by the local agent interactions.
- Efficiency – As *autonomous* entities, agents may provide efficiency by having control of its input, resources and tasks that is independent from a controller. For example, an autonomous Mars rover reduces the communication control overhead from Earth. Agents may also *reason* about how to perform tasks more efficiently. MASs that have *decentralised control* and *adaptivity* provide efficiency by reducing the overhead that would normally be required of a centralised control to adapt and regulate variables in response to changes in the environment. For example, an IDS is efficient in the sense that new attacks are detected by local agents in order to maintain global security.
- Reliability – As *autonomous, reasoning* entities, agents may provide reliability by reasoning about environment changes that may affect task performance. MASs that have *adaptive* behaviour provide reliability in regulating system-level behaviours when the environment changes.
- Robustness – As *autonomous* entities, agents may provide robustness by having control over self-maintaining tasks in the event of a problem (the Mars rover might be capable of self-healing [9]). MASs that have *decentralised control, adaptivity* and *self-maintenance* provide robustness in task operation by adapting to parts (or agents) of the system that malfunction. For example, a battlefield simulation system with decentralised control is robust in operation because it adapts to the loss of sensors.
- Flexibility – As *autonomous, reasoning* entities, agents may provide flexibility by reasoning about new and different tasks to be performed, and having control over when to perform them. MASs may have *adaptivity* that adjusts the system operation to new problems or tasks. For example, the MAS for an IDS provides flexibility in detecting new forms of attack.
- Responsiveness – An agent is *reactive* to environment changes that affect task performance. As an *autonomous* entity, an agent has independent control over how it reacts to environment changes, which improves its responsiveness compared with a centralised controlled approach. MASs may have *decentralised control* that improves the responsiveness of the system due to less overhead in requiring checks with a centralised control.

- Interderminism – As *autonomous, reasoning* entities, agents may reason about what task is best to perform, and have control over the performance of this task.
- Concurrency – MASs as a composition of *autonomous, reasoning* agents may perform tasks that are required to be concurrent.

It is important to note that although we define features independently from one another, often in real world problems they are conjoined. For example, an IDS system requires *responsiveness* in *distributed task* performance in a *dynamic* environment. An agent oriented solution might address these required features with an emergent global *reactive* system response due to *proactive* distributed task performing agents.

4 Framework for Determining MAS Suitability

In this section, we develop a framework that assesses the suitability of a MAS solution given a specific problem. In section 3, we outlined a preliminary set of features that characterise a general set of problems. We also related specific properties of agent-oriented solutions to specific features of problems. Some of the properties of agent-oriented systems (solutions) are due to properties of the agents themselves, and some are due to their collective interactions as a MAS – these are referred to *agent-level* and *system-level* properties respectively [16]. One or more of the agent-level or system-level properties may be useful in addressing the required features for a particular problem. Table 1 illustrates the ratings given (1-5) for each feature on the extent to which it appropriately described the problem domain for a selection of the MAS solutions analysed from the literature. The elicitation of ratings was performed by one of the researchers using an agreed set of instructions that provided very specific definitions for each feature and each rating. An example of the way that we determined the extent to which a problem feature was present is described for robustness:

Robustness – consideration given to the continual performance of the tasks by the software under conditions of adversity (adversity might be computational failure, or communication medium failure).

What consideration is needed towards how robust the software should be?

1. None – the software does not need to be robust
2. Small – a few tasks should be robust in performance
3. Moderate – a number of tasks should be robust in performance
4. Widespread – most tasks should be robust in performance
5. Whole – the whole system should be robust in performance

We constructed a four step framework which takes a problem instance and produces an assessment of the suitability of a MAS solution.

1. Rate the importance of each of the problem and solution-related features identified in Sections 3.1 and 3.2 respectively (and listed in Table 1) for the problem instance.
2. For each feature from step 1, identify one or more properties of a MAS solution that addresses it (see also Section 3.3).

Table 1. Features for alternative problem domains

Feature	Intrusion Detection	Battlefield Simulation	Patient Care	Search & Rescue	Document Recommend	MASFIT
Environ – Dynamic	5	5	4	5	3	4
Environ – Uncertain	4	4	2	4	3	3
Environ – Open	4	4	4	2	2	3
Distributed – Data	5	5	5	5	4	4
Distributed – Resource	2	4	3	4	5	4
Distributed – Tasks	4	2	3	5	4	2
Interact – Negotiation	1	1	1	1	3	5
Interact – Deliberation	1	1	4	1	4	3
Efficiency	3	4	5	4	3	3
Reliability	4	4	4	4	3	4
Robustness	4	5	4	4	5	2
Flexibility	2	2	4	2	5	2
Responsiveness	5	5	3	5	3	3
Indeterminism	4	2	1	4	2	5
Concurrency	4	3	5	2	4	4

(Ratings: 5 very high; 4 high; 3 moderate; 2 low; 1 very low)

Problem domains: Intrusion detection [12]; Battlefield information system [13]; Multi-agent patient care [17]; Human-Robot Teaming for Search and Rescue [18]; Document recommendation tools [19]; and Masfit, fish auction MAS [15]

3. Determine the extent that a MAS solution may address the problem and solution-related features identified in step 1.
4. Using the importance of each feature identified in step 1 and the extent that the MAS solution addresses this feature in step 3, determine the appropriateness adopting a MAS approach to solve the problem.

5 Conclusion and Future Work

Our aim was twofold: firstly to abstract key features from a variety of different problem domains. Features were cyclically pruned on the basis that they were significant determinants of whether or not a MAS was suitable. Secondly, to construct a framework which assesses the suitability of a MAS solution, given a set of problem and solution-related features.

Whilst our feature identification may be biased by the chosen examples and the way each problem was originally framed, resultant features actually generalise (and address) all criteria described by [1, 9, 10] with one exception: when extending legacy systems from [1]. We propose extending our set of features to identify any need for wrapper agents in situations such as extending legacy systems and/or interfacing with other non agent-based resources.

Our framework can be used where alternative ways of addressing a solution are considered. For example in the Mars rover scenario, self-healing might be considered when *robustness* is necessary in performing *distributed tasks* in the event there is a problem. However, self-healing is not the only means of satisfying the need for robustness – an alternative means might be to have redundancy (many of the same system operating). Steps 2-4 in the proposed framework allow the designer to consider the appropriateness of a MAS solution for both these means of satisfying the need for robustness. In addition our framework, being domain independent, may be applied to scenarios in which MASs have yet to be applied in order to determine its suitability.

Our analytic framework requires further work and validation. We currently have a set of suggested problem and solution-related features. We need to validate these features and determine if additional features should be included. We also need to determine the importance of each feature in the overall decision process. We have started by examining the appropriateness of an agent-based solution to satisfying each feature in Section 3.3. As part of our validation process we plan to use the proposed framework on yet untested applications of MAS. We are currently considering call routing and management applications. Further, structured interviews with experts in AOSE will also confirm the appropriateness of the identified features and the extent to which properties of a MAS address identified problem and solution-related features.

References

1. Wooldridge, M.: An Introduction to Multi Agent Systems. Wiley, Chichester (2002)
2. Horlait, E.: Mobile Agents for Telecommunication Applications (Innovative Technology Series: Information Systems and Networks). Kogan Page Science, Portland (2003)
3. Rodriguez, J.A.: On The Design and Construction of Agent-Mediated Electronic Institutions. Artificial Intelligence Research Insitute, UAB - Universitat Autònoma de Barcelona, Barcelona (2003)
4. Guessoum, Z., Rejeb, L., Durand, R.: Using adaptive Multi-Agent Systems to Simulate Economic Models. In: AAMAS 2004. ACM, New York (2004)
5. DeLoach, S.A., Kumar, M.: Multi-Agent Systems Engineering: An Overview and Case Study. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies, pp. 236–276. IDEA Group Publishing (2005)
6. Wooldridge, M., Jennings, N.R., Zambonelli, F.: Multi-Agent Systems as Computational Organizations: The Gaia Methodology. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies, pp. 136–171. IDEA Group Publishing (2005)
7. Padgham, L., Winikoff, M.: Prometheus: A Practical Agent-Oriented Methodology. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies, pp. 107–135. IDEA Group Publishing (2005)
8. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: TROPOS: An Agent Oriented Software Development Methodology. Journal of Autonomous Agents and Multi-Agent Systems 8, 203–236 (2004)
9. De Wolf, T., Holvoet, T.: Towards a full life-cycle methodology for engineering decentralise multi-agent systems. In: Fourth International Workshop on Agent-Oriented Methodologies, San Diego (2005)

10. EUROSCOM: MESSAGE: Methodology for engineering systems of software agents. Final guidelines for the identification of relevant problem areas where agent technology is appropriate. EUROSCOM Project Report P907 (2001)
11. Serugendo, G.D.M., Gleizes, M.-P., Karageorgos, A.: Self-Organisation in multi-agent systems. *The Knowledge Engineering Review* 20, 165–189 (2005)
12. Asaka, M., Okazawa, S., Taguki, A., Goto, S.: A Method of Trading Intruders by Use of Mobile Agents. In: 9th Annual Conference of the Internet Society (1999)
13. Maston, E., DeLoach, S.: An Organization-Based Adaptive Information System for Battle-field Situational Analysis. In: *Integration of Knowledge Intensive Multi-Agent Systems* (2003)
14. Walton, D.: *The New Dialectic* (1998)
15. Cuní, G., Esteva, M., Garcia, P., Puertas, E., Sierra, C., Solchaga, T.: MASFIT: Multi-Agent System for Fish Trading. In: *Proceedings of the 16th European Conference on Artificial Intelligence* (2004)
16. Beydoun, G., Gonzalez-Perez, C., Henderson-Sellers, B., Low, G.C.: Developing and Evaluating a Generic Metamodel for MAS Work Products. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.) *SELMAS 2005*. LNCS, vol. 3914, pp. 126–142. Springer, Heidelberg (2006)
17. Reed, C., Boswell, B., Neville, R.: Multi-agent Patient Representation in Primary Care. In: Miksch, S., Hunter, J., Keravnou, E.T. (eds.) *AIME 2005*. LNCS, vol. 3581, pp. 375–384. Springer, Heidelberg (2005)
18. Nourbakhsh, I., Lewis, M., Sycara, K., Koes, M., Yong, M., Burion, S.: Human-Robot Teaming for Search and Rescue. *IEEE Pervasive Computing* (2005)
19. Pavon, J., Gomez-Sanz, J.: Agent Oriented Software Engineering with INGENIAS. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) *CEEMAS 2003*. LNCS, vol. 2691, pp. 394–403. Springer, Heidelberg (2003)

Multiagent Incremental Learning in Networks

Gauvain Bourgne¹, Amal El Fallah Seghrouchni², Nicolas Maudet¹,
and Henry Soldano³

¹ LAMSADE, Université Paris-Dauphine
Paris 75775 Cedex 16, France

{bourgne,maudet}@lamsade.dauphine.fr

² LIP6, Université Pierre and Marie Curie
104, Avenue du Président Kennedy - 75016 - Paris, France

Amal.Elfallah@lip6.fr

³ LIPN, Université Paris-Nord
93430 Villetaneuse, France
soldano@lipn.univ-paris13.fr

Abstract. This paper investigates incremental multiagent learning in structured networks. Learning examples are incrementally distributed among the agents, and the objective is to build a common hypothesis that is consistent with all the examples present in the system, despite communication constraints. Recently, different mechanisms have been proposed that allow groups of agents to coordinate their hypotheses. Although these mechanisms have been shown to guarantee (theoretically) convergence to globally consistent states of the system, others notions of effectiveness can be considered to assess their quality. Furthermore, this guaranteed property should not come at the price of a great loss of efficiency (for instance a prohibitive communication cost). We explore these questions theoretically and experimentally (using different boolean formulas learning problems).

1 Introduction

Different tasks can be achieved by groups of autonomous agents, either competitively or cooperatively. In this paper, we are interested in multiagent concept learning: group of agents perceive individually and locally the environment (receiving *examples*), and cooperate to come up, each of them, with a “satisfying” underlying concept (*hypothesis*). We believe there are (at least) two features that must be taken into account when a group of distributed agents are collaborating to achieve such a common learning task (we suppose agents to be cooperative and trustworthy):

- the learning task is likely to be *incremental*, that is the training set is not given a priori. Instead examples are collected incrementally (and locally) by agents;
- *communications* between agents cannot be guaranteed. This can be due to the fact that the communication channel may be unreliable, and/or simply that there exists a (fixed or evolving) topology constraining interactions between agents. In this paper we are concerned with this latter aspect of the problem.

A typical example application would be a distributed sensor network, where each sensor (seen as an agent) would have control over some restricted area, and would need

to interact with neighbours sensors in order to analyze the behaviour of some unknown third-party. The research challenge is then to design mechanisms that enjoy efficiency properties without being overwhelmingly burdening as far as communication is concerned. We are not aware of any attempt to tackle both features simultaneously. In particular, an incremental multiagent algorithm was proposed in [1], but it assumes that the system is a fully connected society of agents. More generally, learning in complex situations (involving uncertainty and unreliable communications) is performed with a numerical model, see *e.g.* [2]. Ontañón *et al.* [3] is a notable exception, and presents an argumentation based multiagent learning system for case-based reasoning.

The purpose of this paper is to provide a first approach to tackle symbolic learning in networks of agents (as we shall see, our experimental study provides interesting clues to develop more refined variants). This work will be based on a hypothesis refinement framework described in [4]: the general idea of this mechanism is to allow agents to propagate relevant examples to a learning agent. While the proposed mechanism has been shown to guarantee some theoretical consistency properties, other aspects should be considered before one can conclude that it is practically useful. For instance, it is crucial to analyze how costly would be such a mechanism in terms of communications. The first contribution of this work is to provide such an analysis. In practice, there are many other parameters that are needed to assess the quality of the mechanism. Is the obtained hypothesis accurate, and concise? Does the mechanism end up replicating every examples? Does it requires lots of internal computations? In this paper, we discuss these aspects by performing a number of experiments on different networks representing typical structures (like lines, trees, or small worlds).

The remainder of this paper is as follows. Section 2 presents the basic ingredients of our approach, giving details on the knowledge representation, and the key notion of consistency, which represents the adequacy between an hypothesis and a set of examples. Section 3 gives the details of the main mechanisms used in the paper. The proposed mechanism can be guaranteed to ensure consistency at the global level, but may require a quadratic number of communications in the worst-case. To get a clearer picture of the situation, we ran several experiments that are reported and commented in Section 4. Section 5 discusses possible extensions, and Section 6 concludes.

2 Formal Model

This section presents a general formal model for dealing with consistency between groups of agents in a multiagent system. It is here presented in the context of multiagent concept learning (interpreted as the building of a common consistent hypothesis).

2.1 Knowledge Representation

We take a *system* populated by n agents a_1, \dots, a_n . Each agent a_i has two different kinds of knowledge:

- E_i is the *example memory*, representing *certain, non-revisable* individual knowledge of each agent. This set represents the *collected knowledge* of each individual agent, and would be called *observation set* in other contexts. These examples are *certain* and *fully described* (we do not consider uncomplete or noisy information).

- h_i is the *working hypothesis*, or *favourite hypothesis* of the agent, representing the *uncertain*, explicit knowledge of the agent. It is derived from non-monotonic inferences and is therefore *revisable*.

The following representation will be used for learning boolean formulae. We consider a propositional language \mathcal{L}_p , defined over a set of atoms \mathcal{A} . An example e in an example memory is represented by a tag $+$ or $-$ (depending whether they belong or not to the target concept) and a description which is a *conjunctive statement*, that is, a conjunction of atoms from \mathcal{A} . An example memory E_i will then be divided in two sets E_i^+ and E_i^- that contains respectively examples labeled with $+$ (positive examples) or $-$ (negative examples). An *hypothesis* will then be a disjunction of conjunctive statements, or *terms*, that is $h_i = p_1 \vee \dots \vee p_m$, where each term p_k is a conjunction of atoms ($p_k = a_1 \wedge \dots \wedge a_l$) from \mathcal{A} . A hypothesis is thus a formula in disjunctive normal form.

2.2 Consistency

As hypotheses are uncertain, they can be in contradiction with some other knowledge. We want to ensure that the working hypothesis h_i has some property ensuring its internal coherence and adequacy with the non-revisable knowledge E_i . Very abstractly, we shall capture this adequacy by a complete binary relation $Cons(h, E)$ between an hypothesis h and certain knowledge E , the *consistency relation*. For a given case, we will assume that there exists a hypothesis h_0 that is consistent with the union of the sets of all example memories (in our concept learning application, it means that the target concept can be expressed in the hypothesis language). To instantiate consistency in concept learning, we first introduce a *generality* relation. We will say that a term p is *more general* than another term p' ($p \models_s p'$), iff the constituting atoms of p are included in those of p' . Then a hypothesis *covers* an example iff one of its term is more general than the example's description, and covers an example set iff it covers every examples in it. We shall then consider that an hypothesis is *consistent* with a set of examples E (noted $Cons(h, E)$) iff it ensures (i) *coherence*: h does not cover any negative example of E^- ; and (ii) *completeness*: h does cover all positive example of E^+ . This consistency relation is *compositional*, meaning that: $Cons(h, O)$ and $Cons(h, O')$ iff $Cons(h, O \cup O')$.

This notion of consistency is easily extended to groups of agents.

Definition 1 (Group Consistency). *An agent a_i or an hypothesis h_i is group consistent wrt. the group of agents G ($GCons(a_i, G)$ or $GCons(h_i, G)$) iff $Cons(h_i, \cup_{a_i \in G} E_i)$.*

Depending on the cardinality of the group G , we can talk of different “levels” of consistency: (i) *internal consistency*— the limit case when G is limited to a single agent (ensures that its hypothesis is consistent with its certain knowledge ($Cons(h_i, E_i)$)); (ii) *peer consistency*— when G is a pair of agents (important in our context for characterizing results of local bilateral communications.); and (iii) *MAS-consistency*—the limit case involving all agents within the society.

2.3 Revision Mechanisms

To ensure consistency at these different levels, agents will use reasoning and communication processes that we shall call *revision mechanisms*. As with consistency, it is

possible to consider different levels of revision mechanism depending on the number of agents involved in the process. To begin with, each agent will be equipped with an *internal revision mechanism*, which is a mechanism μ by which an agent a_i (with its working hypothesis h_i and its example memory E_i) receiving an example e updates its example memory by adding up e , and update its working hypothesis to $h'_i = \mu(E_i, h_i, e)$ such that $Cons(h', E_i \cup \{e\})$ (such a mechanism is said to preserve internal consistency). Then, a *local revision mechanism* is a mechanism \mathcal{M}^2 by which an agent a_i receiving a labeled example e communicates with another agent a_j to update its working hypothesis and possibly the hypothesis of the other agent. A *global revision mechanism* is a mechanism \mathcal{M}^n by which an agent a_i receiving a labeled example e triggers a series of local revision mechanisms to update its working hypothesis and possibly the hypotheses of the other agents. A revision mechanism \mathcal{M} is said to guarantee $GCons(a_i, G)$ iff, for any example e reaching a_i , it is the case that the execution of \mathcal{M} by a_i with G will result in a situation where $GCons(a_i, G)$ holds.

2.4 Communicational Constraints

The communication of agents is restricted by topological constraints. A given agent will only be able to communicate with a number of *neighbours*. We can then construct a communication graph representing these communication links between the agents [1]. In the following, we will suppose that this graph is *connected*, that is, it contains a path between any pair of agents, and that it remains *static* during the time required for the revision process (though it could possibly change between revisions).

2.5 Problem Description

Given a system of n agents a_1, \dots, a_n , the objective is usually to devise some global mechanism guaranteeing MAS-consistency. Agents cooperate during the revision to form a *common* MAS-consistent hypothesis. When an agent a_i receives a new example contradicting the common hypothesis, this agent triggers a revision mechanism to rebuild another hypothesis and ensure its MAS-consistency. Once the revision is done, another example can be received, and the process can be iterated. This iteration produces a sound multiagent incremental learning process. The multiagent incremental learning mechanism described in [11] can be interpreted as a global revision mechanism guaranteeing MAS-consistency in fully connected societies of agent. However, this solution offers no guarantee when we have to deal with some communicationnal constraints restricting the communicationnal links between the agents.

3 Mechanisms with Propagation

In the fully connected context studied in [11], a single *learner* agent initiating the revision would use in turn all the other agents as *critics* to validate its hypothesis until

¹ We assume that the relation that links two neighbours in a communication graph is symmetric, but of course not transitive. Symmetry ensures that an agent can always reply to a request.

all these agents would directly accept the same hypothesis. However, when communications are constrained, a single learner cannot directly propose its hypothesis to all others agents. The global mechanism must then rely on some kind of propagation. In [4], a layered mechanism based on these principles and decomposed at three levels has been proposed. It is a simple mechanism that we study here as a first step-stone whose analysis will be helpful to devise more refined mechanisms in the future. Each agent possesses an internal revision mechanism μ preserving internal consistency. Two agents can then reach a common peer-consistent hypothesis through a local revision mechanism where a *learner* agent use its internal revision mechanism to propose hypotheses to a *critic* agents. At last, a global revision mechanism articulates these local exchanges to ensure that MAS-consistency is reached. We first recall briefly the basic ideas of this mechanism, before turning on to the examination of its properties.

3.1 Learning Process

Our internal revision process is based on MGI, an incremental bottom-up learning process fully described in [5]. As said before, the hypothesis is a disjunction of terms p . In MGI, each of these terms is by construction the *lgg* (least general generalisation) of a subset of positives examples $\{e_1, \dots, e_n\}$, that is the most specific term covering $\{e_1, \dots, e_n\}$. The *lgg* operator is defined by considering examples as terms (using their description), so we denote as $lgg(e)$ the most specific term that covers e , and as $lgg(h, e)$ the most specific term which is more general than h and that covers e . Restricting the term to *lgg* is at the core of many bottom-up learning algorithms (for instance [6]). Taking a current hypothesis h , consistent with an example memory $E = E^+ \cup E^-$, and a new labeled example e , the revision mechanism builds a new hypothesis h' consistent with $E \cup \{e\}$. There are three possible cases: *consistent example*— $Cons(h, \{e\})$ (no update needed); *positive counterexample* — e positive and $h \not\models_s e$ (here, we seek to generalise in turn the terms p of h . If a coherent generalisation $p' = lgg(p, e)$ is found, p' replaces p in h , otherwise $h \vee lgg(e)$ replaces h); and *negative counterexample* — e negative and $h \models_s e$ (each term p covering e is then discarded from h and replaced by a set of terms $\{p'_1, \dots, p'_n\}$ that is, as a whole, coherent with $E^- \cup \{e\}$ and that covers the examples of E^+ uncovered by the suppression of p).

3.2 Local Hypotheses Exchange

At the local level, pairs of agents use a simple protocol for hypothesis exchanges, as proposed in [4]. The agent applying this mechanism, called \mathcal{M}_{ij}^2 , takes an active role in building and refining an hypothesis. It is called the *learner* agent. The second agent is a *critic*, that uses its knowledge to acknowledge or invalidate the proposed hypothesis. The learner agent a_i first updates its hypothesis h_i to h'_i using an internal revision mechanism μ guaranteeing internal consistency. Then it *proposes* it to its partner agent a_j , and a_j either replies with *accept_{direct}* and adopts h'_i as its new working hypothesis if $Cons(h'_i, O_j)$, or otherwise sends *counter-example(σ')*, where $\sigma' \in O_j$ is such that $Cons(h'_i, \{\sigma'\})$ is false. Upon reception of a counter-example, a_i applies again μ to revise its hypothesis with the new observation, and proposes the resulting hypothesis as before, except that an acceptance will now result in a *accept_{indirect}* message. This

revision mechanism can be shown to guarantee *peer consistency* [4], and ensures that afterwards, both agents share the *same* hypothesis.

3.3 Propagating Hypotheses in a Network

In [4], Bourgne *et al.* propose to use the same principle as in [1] of a main *learner* being the origin of the hypothesis, but neighbours will be in turn critics and learners, acting as proxies to propagate the hypothesis. To ensure that propagation does not reach the same agent from two different ways, cycles are eliminated by constructing (on-the-fly) a spanning tree (a tree-like sub-graph that contains all the nodes of the original graph). The global revision mechanism \mathcal{M}_P^n [4] does this while propagating the hypothesis. The agent initiating \mathcal{M}_P^n is called the root, it will have a role of *learner* during the whole process, and will be the source of the adopted hypothesis. It is initially the only marked agent. It first sends *request-links* messages to all its neighbours, who become its children and are thus marked. Then, it selects its first child to take a role of *critic* in a local exchange (with $\mathcal{M}_{U_j}^2$), before asking it to propagate the resulting hypothesis. When asked to propagate an hypothesis, a child a_c would first send *request-links* to its neighbours (except its parent) if it has not already done so (marked neighbours will reject the request while others would become its children).

- If some child has not accepted its current hypothesis yet, a_c would select it to become a critic and take the role of the learner to propose in turn its hypothesis (adopted from its parent).
 - If the local exchange ends with *accept_{direct}*, a_c asks its child to propagate in turn this hypothesis.
 - If it ends with *accept_{indirect}*, as a_c has learned a counter-example and changed its hypothesis, it informs its parent by sending a message *hyp-changed*. Receiving this message, the parent will act as a learner to get the new counter-example through a local exchange, and warns its own parent to do the same until the root get the counter-example, and build a new hypothesis.
- Otherwise, all its children, if any, have acknowledged its hypothesis h , and a_c can tell its parent that it acknowledges h (*strong-accept*). The parent would then check if it has other children that have not yet strongly accepted h , and start a local exchange with them or send a *strong accept* to its own parent.

When the root has received a *strong accept* from all its children for its current hypothesis, it means that this hypothesis is consistent with all agent and has been adopted throughout the whole network. The global revision is completed. This mechanism guarantees MAS-consistency (provided the system is connected, of course), and the resulting MAS-consistent hypothesis is shared by all agents in the system after termination. Note also that it behaves like the SMILE algorithm [1] when the network is fully connected. However, this only provides a rather coarse-grained understanding of the effectiveness of this mechanism, and many questions remain. How accurate is the obtained hypothesis? How compact is it? On top of that, the complexity of the process consisting in building the propagation network on-the-fly while propagating the hypothesis may raise doubts on its practical relevance. If the communication cost involved is prohibitive, this would be a severe weakness if the mechanism were to be deployed on networks involving a large number of agents. We now turn our attention to these issues.

3.4 Communication Complexity

To discuss the communication complexity of our mechanism, we distinguish two types of communications, whether their purpose is to build the spanning tree (*link requests*) or to validate a hypothesis. The first kind of communication can indeed be done once for all before learning if the communicational links are stable, as we assume here. For establishing the spanning tree, each agent has to request links with each of its neighbours (except its parent, if any), the cost would thus be $\mathcal{O}(e)$ where e is the number of edges. The worst case would then be $\mathcal{O}(n^2)$, when the graph is fully connected.

Then, for validating a hypothesis, we first compute the number of messages that is needed to propagate a counter-example back to the root in the worst case. This occurs when the counter-example is as far as possible from the root, and is used as a critic in last position. Let L be this maximal path between the root a_0 and an agent a_k . The hypothesis of the root would only be proposed to a_k after being validated by all the other agents (as a_k is the last critic), thus after $n - 1$ series of dialog which size is roughly constant (*propose(h)*, *accept*, *propagate*, and later on *strong-accept*). Then, when the hypothesis is proposed to a_k , the counter example is given, and a series of L exchanges (namely, *propose(h)*, *counterexample(e)*, *propose(h')*), *accept*, *hyp-changed* propagate this counter-example back to the root. Thus, for each counter-example, we have $\mathcal{O}(L + n)$ communications of 4 to 5 messages. As in the worst case, $L = n - 1$, there is at worst $\mathcal{O}(n)$ communications per counter-example. The total number of communications for hypothesis validation is thus $\mathcal{O}(nc)$ where c is the number of counter-examples involved in building the hypothesis, which surely cannot be greater than the number of examples in the system that are not known by the root. This upper bound on the number of counter-examples involved is tight, as the following example shows.

Example 1. Suppose n agents are connected on a single line, from a_1 to a_n . The concept to be found is $ab \vee oc$. The example memories are initially empty, except for E_n which contains two examples $\{bac+, bec-\}$. The working hypothesis is bac for all agents. Now a_1 receives $boc+$: h is made more general and becomes bc . This hypothesis is propagated and accepted by all agents until a_n , which provides the counter-example $bec-$. Then a_{n-1} proposes an empty hypothesis, since its memory does not contain any positive example to cover. a_n provides again a counter-example by exhibiting $bac+$: a new hypothesis bac is formed and accepted. But now observe that the same dialogue will bilaterally occur between agents to propagate back to the root, where $boc \vee bac$ is accepted. In this example, the two examples initially present in the system have been used by each agents as counter-examples.

In practice, c will really depend on n . Indeed, the probability that an hypothesis incrementally built by an agent is contradicted by another example in the system depends on the number of examples upon which it is built, and such a number will tend to be lower when n increases (as each agent holds fewer examples).

4 Experiments

We now describe our experiments with this mechanism, used for learning several difficult boolean formulas. An experiment is typically composed of 20 to 50 runs, each

one corresponding to a sequence of examples incrementally learned by the MAS: a new example is sent to a random agent when MAS-consistency is restored.

4.1 Parameters

We considered *different boolean problems* of various difficulty. The 11-multiplexer (M11, see [7]) uses 3 address boolean attributes a_0, a_1, a_2 and 8 data boolean attributes d_0, \dots, d_7 . The formula is satisfied when the number coded by the 3 address attributes is the number of a data attribute whose value is 1. More complex variants are M11_9, where 9 irrelevant attributes are added to the examples description, and the 20-multiplexer, M20, which involves 4 address boolean attributes and 16 data boolean attributes. X-OR problems encodes a number of boolean attributes whose sum must be odd. We will use these parity problems with 5 attributes, and either 5 or 15 irrelevant attributes (XOR5_5 and XOR5_15). At last, we shall use a simple DNF with 4 terms on 9 attributes, DNF4-9, whose formulae is $a_0a_1a_2 \vee a_2a_3a_4 \vee a_4a_5a_6 \vee a_6a_7a_8$. The total number of attributes, and the number of terms of the target formulas of these problems can be found in Table 1. Note that there are 2^p possible examples where p is the number of attributes, and half of them are positive (except in DNF4-9).

We tested the mechanism in different settings. Two parameters can vary: the number of agents in the system, and the type of network. Different classical topologies of networks have been tested. *Fully connected graphs* are used to compare the performance of \mathcal{M}_p^n with those of SMILE [1] in an equivalent context. *Lines graph*, where every agent a_i is connected to two agents, a_{i-1} and a_{i+1} except for a_0 and a_n . It typically provides worst-case scenarios. *Regular trees* of degree 3 are taken as examples of hierarchical structures. Finally, we used *small worlds*, that are quite common in self organised networks such as for instance peer-to-peer or social networks.

4.2 Evaluation

We assess the performances according to several effectiveness and efficiency measures.

- *Effectiveness* is evaluated through two different measures. *Accuracy* is the classical rate of accurate classification of a batch of test examples distinct from the learning set, whereas the *hypothesis size* is the number of terms in the hypothesis. A small hypothesis is more simple and easier to understand.
- *Efficiency* is concerned with three aspects: (i) *redundancy*, which is used to study the distribution of the examples in the MAS memory, is written $R_S = n_S/n_e$, where n_S is the total number of examples stored in the MAS, and n_e is the total number of examples received from the environment in the MAS; (ii) *computational cost* is the mean number of internal revisions performed during a global revision; and (iii) *communicational cost* finally is mainly measured through the mean size of data transfer per global revision, which expresses the cost in bandwidth of the communications during a revision.

4.3 Results

We only illustrate the main results of our experiments with M11 and XOR5_15 problem, but these conclusions on the general behaviour have been checked with all problems.

Redundancy and computational cost. Figure 1 shows the redundancy results for different 20 agents networks faced with a M11 problem. We can see that for fully connected societies, the propagation mechanism behaves exactly like SMILE, as ensured by construction. In other topologies, the redundancy is slightly higher, especially for lines, but remains quite low (e.g. 2.85 for M11 with a line of 20 agents). The average ratio of redundancy between \mathcal{M}_P^R and Smile is also low (around 1.2), with a maximum of 1.66 for lines of 20 agents in complex problems (XOR5_15). It tends to be more important in complex problems, where redundancy is slightly higher. Dealing with communicational constraints might increase redundancy, but not importantly. The same behaviour is observed for the computational cost. In all cases, the number of internal revision per global revision and the final redundancy appear to be linear with the number of agents. Besides, the shorter the Characteristic Path Length (CPL) in the spanning tree, the better the redundancy and computational cost.

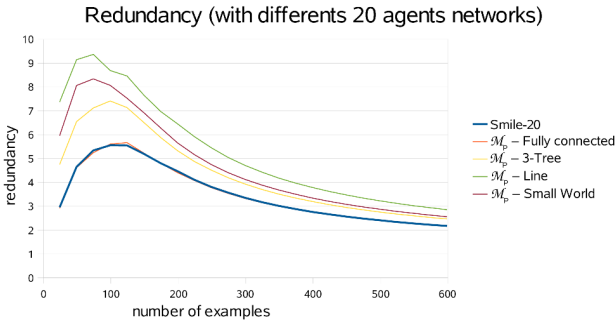


Fig. 1. Redundancy of \mathcal{M}_P with 20 agents in different topologies compared to a 20-SMA using Smile (M11 problem)

Communicational cost. We now detail the results regarding communicational efficiency. Figure 2 gives the mean size (in bits) of data transfer per global revision for M11 problem. The propagation mechanism again has a slightly higher cost than SMILE. But here, due to the linking process, the worst configuration is the fully connected society. In order to ensure a better readability of the results, we used static structures, but as the spanning tree is rebuild at each revision, these results should be the same with variable connections. However, if we know that the links are static, we can build the spanning trees only once and memorize it. To assess communicational cost in such cases, we measured a *link free communicational cost* that does not take into account the linking process. As a result, all topologies have more similar cost, though the line network still suffer from the highest number of exchanges needed to get a counter-example back to the root.

Effectiveness. Accuracy and hypothesis size have been measured for different topologies with our mechanism and compared with results from SMILE [1]. Similarly, our mechanism improves both accuracy and hypothesis size when compared to a single agent using MGI. This result holds for *all* topologies tested, however the extent of

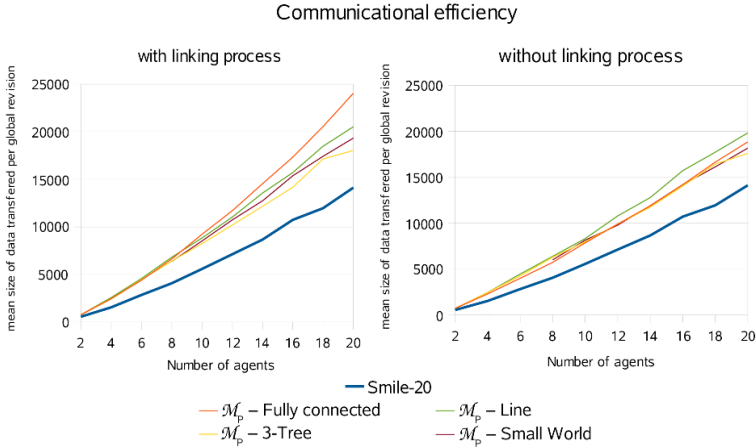


Fig. 2. Communicational cost of \mathcal{M}_P in different topologies compared to a 20-SMA using Smile, with or without linking process (M11 problem)

this gain varies: best results are observed for fully connected societies, whereas lines structured societies of agents get less benefit. This accuracy gain is observed for all our boolean problems, especially for those whose difficulty is related to the size of the hypotheses space. Having each new hypothesis formed over only a subset of the information in the system and then refined through relevant counter examples seems to guide the exploration of the hypotheses space towards better hypotheses. Our internal learning process is indeed order dependent. Trying several combinations depending on the agents forming the hypotheses and having relevant examples then given in order boost the effectiveness of this bottom-up algorithms. Note that though there are several agents and memories implied, this is not an “ensemble effect” as a single common consistent hypothesis is formed. Topologies in which a given agent has more neighbours in its spanning tree show significantly better results. Our conjecture is that it is linked with redundancy. The more an hypothesis is refined by relevant counter-examples, the more precise it grows. Hypotheses formed with a small number of examples are more likely to be refined. If redundancy is low, each agent will only have a small fraction of the total number of examples, and thus hypotheses formed individually by them are more likely to trigger refinement. This conjecture is supported by experimental measure showing that the most precise hypotheses correspond to the maximal number of internal revisions. Better results are observed on topologies that exhibit low redundancy (that is fully connected graphs and 3-trees).

This result also holds for mean hypothesis size, and is verified for our different boolean problems, as shown in the following table, giving accuracy results of \mathcal{M}_P with trees, lines or small worlds of 20 agents, along with the results of SMILE with a single agent and a 10 agents MAS, and those of two standard algorithms: JRip (an implementation of RIPPER [8]) and Id3 [9]. Table 1 summarizes the results.

² For the experiments with JRip and Id3, we measured the mean accuracy on 50 trials. JRip and Id3 parameters are default parameters, except that JRip is used without pruning.

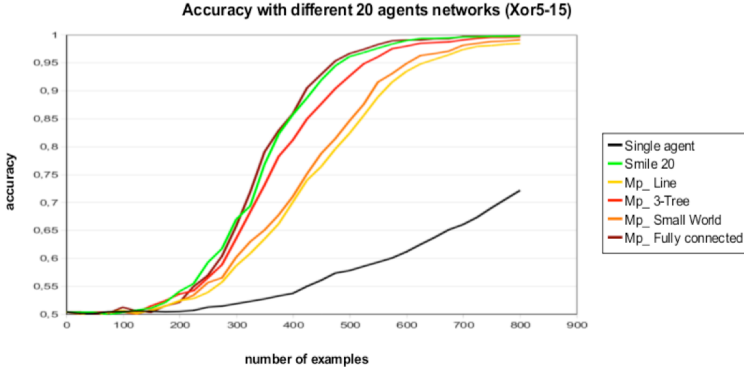


Fig. 3. Accuracy of \mathcal{M}_P with 20 agents in different topologies compared to a 20-SMA using Smile and a single agent 9XOR5_15 problem)

Table 1. Characteristics and compared accuracy results of different boolean problems

Pb	att.	terms	learn.	set	JRip	Id3.	Sm 1	Sm 20	\mathcal{M}_P -3tree 20	\mathcal{M}_P -line 20	\mathcal{M}_P -sw 20
M11	11	8	200	88.3	80.7	88.7	96.5	95.2	93.8	96.5	
M11_9	20	8	200	73.4	67.9	66.8	82.7	80.5	72.3	77.5	
M20	20	16	450	67.7	62.7	64.6	83.0	82.6	74.3	76.4	
XOR5_5	10	16	180	52.6	60.8	71.1	83.4	82.6	80.3	81.8	
XOR5_15	20	16	500		58.7	95.9	92.6	82.4	84.6		
DNF4-9	9	4	100		89.2	95.5	94.5	94.8	94.7		

5 Extensions

From our experimental results, it is clear that accuracy, redundancy, computational cost and (link free) communicational cost depend on the structure of the networks. More precisely, results are better when the spanning trees used for propagation have a small depth. It is due to the fact that bringing back counter-examples to the root is quite costly. Moreover, as each link in the process gets the counter-example, it augments redundancy, and thus impact effectiveness. Improved versions of this mechanism should then seek to diminish as much as possible this effect. A first idea is to improve the adequacy of our spanning tree, by ensuring that it is not deep. If we use a different spanning tree for each possible root, or build it dynamically, the minimum depth can be achieved by building it width-first. However, if one wants to use a single spanning tree for all updates, then we should seek to minimize its mean depth for each possible root, which is equivalent to minimizing its CPL. Another method would be to avoid tracking the counter examples back to the root by changing root every time that an hypothesis is refined. The agent who refined it would thus become a new root and proposes its hypothesis to its children. To avoid sending too much messages for building new spanning trees, it would then be best to just adapt the former tree by just changing its root and reversing some parent-child links. A last idea is to have agents drop the examples they learned from others after each global revision. This would ensure that there is no redundancy in the system.

First results with XOR5_15 shows that accuracy is indeed improved (98% accuracy with 500 examples in MAS of 20 agents), and no more dependent of the network structure. However, communicational cost is then much more important and still depends on the topology. Some tradeoff (dropping only part of the external examples) could be useful if the communicational cost is important.

6 Conclusion

This paper provides an analysis of a general consistency revision mechanism dealing with communicational constraints in structured networks. We have discussed its communicational complexity, and evaluated its effectiveness and efficiency in a multiagent concept learning application, where it has been compared to a simpler mechanism for fully connected multiagent systems. Experimental results have shown that our mechanism is as effective and only slightly less efficient, despite these constraints. Finally, we have presented some leads to improve it, based on the analysis of experimental results.

References

1. Bourgne, G., Seghrouchni, A.E.F., Soldano, H.: SMILE: Sound Multi-agent Incremental LEarning. In: Proc. of AAMAS 2007, pp. 164–171. ACM Press, New York (2007)
2. Stone, P.: Intelligent Autonomous Robotics: A Robot Soccer Case Study. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, San Francisco (2007)
3. Ontañón, S., Plaza, E.: Recycling data for multi-agent learning. In: ICML 2005, pp. 633–640. ACM Press, New York (2005)
4. Bourgne, G., Seghrouchni, A.E.F., Maudet, N.: Towards refinement of abductive or inductive hypothesis through propagation (2008) (under review)
5. Henniche, M.: MGI: an incremental bottom-up algorithm. In: IEEE Aust. and New Zealand Conference on Intelligent Information Systems, pp. 347–351 (1994)
6. Califf, M.E., Mooney, R.J.: Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Mach. Learn. Res.* 4, 177–210 (2003)
7. Esmeir, S., Markovitch, S.: Lookahead-based algorithms for anytime induction of decision trees. In: ICML2004, pp. 257–264. Morgan Kaufmann, San Francisco (2004)
8. Cohen, W.W.: Fast effective rule induction. In: ICML, pp. 115–123 (1995)
9. Quinlan, J.R.: Induction of decision trees. *Machine Learning* 1(1), 81–106 (1986)

UML-F in the Design of an Agent-Oriented Software Framework

Daniel Cabrera-Paniagua and Claudio Cubillos

Pontificia Universidad Católica de Valparaíso, Escuela de Ingeniería Informática,
Av. Brasil 2241, Valparaíso, Chile
daniel.cabrerap@gmail.com, claudio.cubillos@ucv.cl

Abstract. This paper presents a practical development of a study case regarding the utilization of a non-standard UML-F profile for the specification and documentation of an agent-oriented software framework devoted to the passenger transportation domain. UML-F profile does not pretend to replace the standard UML, but to use existing elements in it to represent in a better way the reality of a software framework through the adaptation of UML artifacts. The PASSI methodology has been used as base software engineering methodology for leveraging the multiagent architecture. As a way of ensuring proper development, it has been complemented with a general process for framework development.

Keywords: UML-F, Framework, Agents, Passenger Transportation.

1 Introduction

Recognizing in the software reuse an alternative with many advantages, such as ensuring lower development time and lower costs, increased quality levels, increased productivity, decreased difficulties at the maintenance phase and support in software production, among the most important, it is necessary to include the techniques and procedures in developing software aimed at promoting the software reuse. One of these elements corresponds to the way which a reusable system is described and documented. The non-standar profile UML-F (UML-Framework) [7] supports the development and adaptation of object-oriented frameworks, focusing on the location of the framework variability points.

The current work presents the design of an agent-oriented software framework for the passenger transportation domain, considering the UML-F profile for the framework specification. Particulary, in the domain of passenger transportation, the obtaining of a software framework for passenger transportation will allow to adapt this architecture according to concrete required characteristics, reducing in an important way the development cost, and not less important, while using part of a solution that already has been used in the past and therefore with the necessary maturity and reliability.

The framework development has been realized on the basis of the use of PASSI [1], a multiagent-oriented methodology for systems development. PASSI integrates design models and concepts from both OO software engineering and artificial intelligence approaches using the UML notation. As a way of ensuring proper development, it has been considered the use of a general process for framework development along

with PASSI, as the latter is designed for concrete systems rather than frameworks. The considered process for framework development corresponds to the one presented in [3].

This work represents the continuity of a past research in this transportation domain [6], concerning the development of an agent system for passenger transportation for a single operator under a demand-responsive scenario.

2 Related Work

In the field of systems development, once the different requirements to satisfy are relatively stabilized, it is necessary to express the characteristics which will be the future system through a high-level specification language or using a graphic convention. At present, the standard UML is one of the most widely used in the specification, visualization and documentation requirements. However, UML is not exactly an Architecture Description Language (ADL). An ADL is a language that delivers elements to model the conceptual architecture of a software system, modeling explicitly the components, connectors, and their configurations [2]. However, in this work has been used UML, because the modeling architecture has been carried out following the steps of PASSI methodology, and using a graphical tool called PASSI Toolkit [4], based on UML artifacts.

Is possible to indicate that both, the PASSI methodology and the PASSI Toolkit were designed for development of agents systems, and not precisely for the developing of agent-oriented software frameworks, situation observed in this paper. By the same reason, the UML-F profile was used in the development of some artifacts, with the aim to fill this lack, and somehow verify the applicability of UML-F profile in the field of software agents.

3 Software Frameworks and UML-F

According to [5], *“a framework is a set of classes that embodies an abstract design for solutions to a family of related problems, and supports reuses at a larger granularity than classes”*.

The UML-F profile constitutes an important aid in the specification and documentation of software frameworks, as it covers and formalises those aspects not covered so far by the UML standard with regard to the area of modeling frameworks. Some of the most important features of the UML-F profile are: provides elements of notation to adequately document well-known design patterns; is built on the standard UML, that is, the extensions generated can be defined on the basis of mechanisms extension which in UML already exist; is, in itself, the medium that allows direct way to document any pattern framework.

4 Multi-agent Framework Design

In this section, agent framework artefacts are depicted following the PASSI steps. The first diagram presented is the Domain Description Diagram (see Figure 1). This

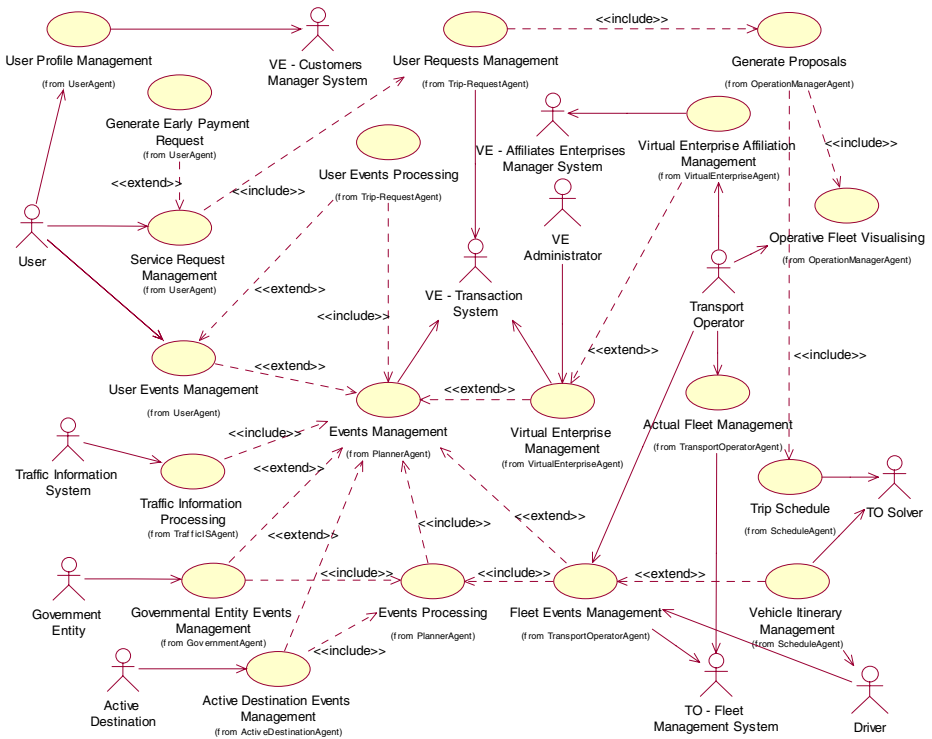


Fig. 1. Domain Description Diagram

diagram is framed within the first stage of the PASSI methodology, corresponding to the System Requirements Model, and is based on UML use-cases, offering a general view of all the functionality provided by the system.

Figure 2 shows a Roles Identification Diagram, in which a transport operator wishes to affiliate to the virtual transportation enterprise by sending a request of affiliation. This request, administered by the VirtualEnterpriseAgent, is verified in the fulfillment of norms, internal policies of the virtual transportation enterprise, and the business opportunities that this affiliation represents. Having made these checks, the agent returns a final response on the request.

After knowing the final answer, there is an alternative course of action, represented by a horizontal labelled bar that is located along the diagram. This label indicates the existing courses of action, in this case, if the request is accepted or is rejected. Each course of action alternative has an independent sequence diagram, which describes the course of action and messaging agreement existing with the final decision taken at the virtual transportation enterprise. If the affiliation request is accepted, the transport operator orders the activation of its vehicle fleet. This order is managed by the agent TransportOperatorAgent, who is responsible for such a change to take place within its Fleet Management System. After this, vehicles are available within the virtual transportation enterprise to participate in the fulfilment of transportation requests. On the

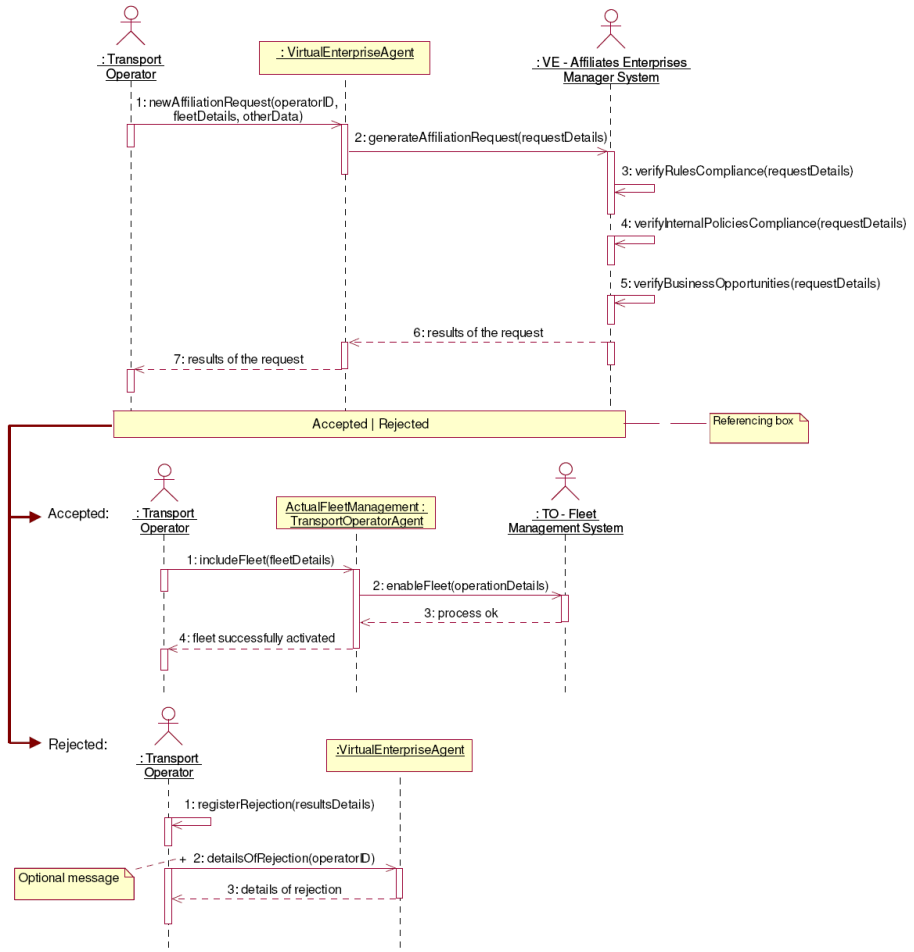


Fig. 2. Roles Identification for the scenario: “Transport Operator sends an affiliation request to the Virtual Transportation Enterprise”

other hand, if the affiliation request is rejected, the transport operator registers its request rejection. Eventually, may request further details on the request rejection causes. This optional message is graphically represented using the "+" symbol, which indicates "optional messaging" within the UML-F profile.

The Figure 3 shows a diagram of the phase of Agent Implementation Model, which is the Multiagent Structure Definition. It is possible to view all actors within the defined architecture in development, and its relationship with the various agents, and the identifying of transactions related to each of them. The classes identified in the figure with the symbol "..." indicate that have not yet been established all internal elements of them (both attributes and methods).

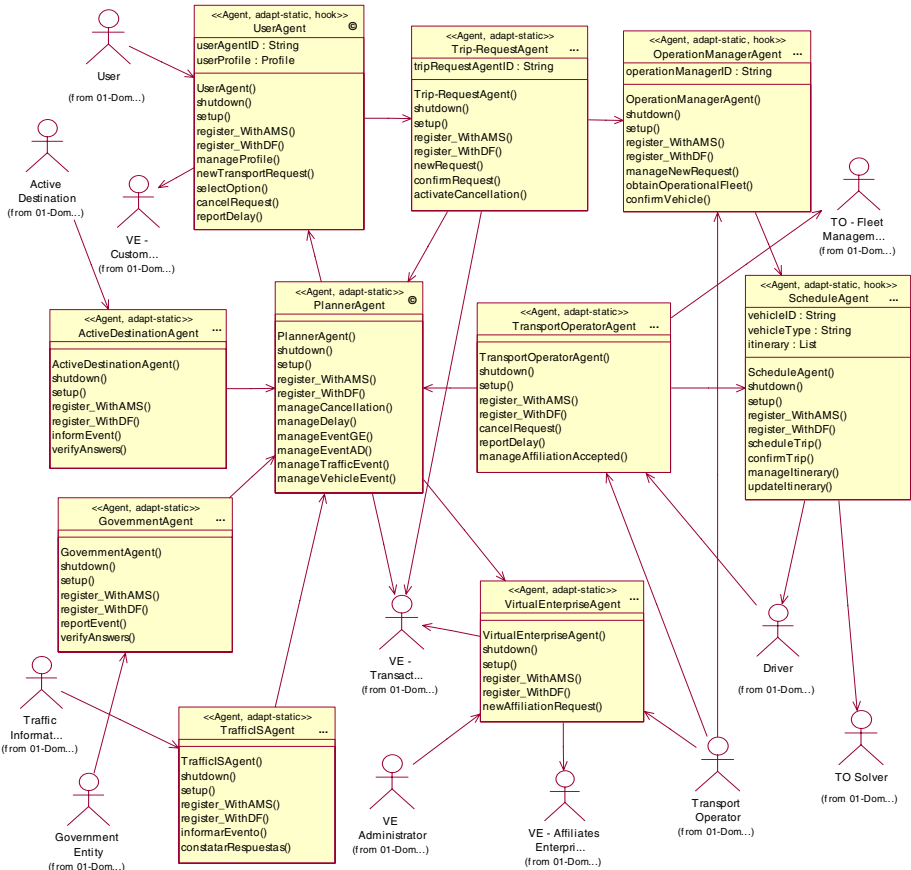


Fig. 3. Multiagent Structure Definition Diagram

On the other hand, the classes with the symbol "©" are those whose methods and attributes shown are actually all the ones the class possesses.

The stereotype <<agent>> indicates that the classes are agents, and the stereotype <<adap-static>> denotes those classes that may be subject to changes, but only changes at the design phase (at runtime is not possible to observe changes in its internal structure). The stereotype <<hook>> indicates that the class has at least one method of type "hot spot", that is, their characteristics depends on each particular implementation derived from the model defined.

5 Conclusion

The specification and documentation of an agent-oriented software framework using the UML-F profile has been achieved. The incorporation of UML-F into PASSI artifacts promotes a wider understanding of the design (as it is UML-based) while increasing the original expressiveness of PASSI for making explicit the variable points

in the resulting agent framework. This software framework will lay the foundations for the future development of multiple systems independent from it, providing a core of basic functionality for the passenger transportation, and providing flexibility to adapt the architecture to diverse concrete situations.

A functional prototype has been developed by extending the original framework and is currently being tested with Solomon's benchmark data sets for VRP and specially adapted for the passenger transportation problem.

Acknowledgements. Partially funded by Grant PUCV 037.215/2008 under the "Collaborative Systems" Nucleus Project.

References

1. Burrafato, P., Cossentino, M.: Designing a Multi-Agent Solution for a Bookstore with the Passi Methodology. In: Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems, pp. 102–118 (2002)
2. Vestal, S.: A cursory Overview and Comparison of Four Architecture Description Languages. Technical report, Honeywell Technology Center (1993)
3. Mattsson, M.: Object-Oriented Frameworks: A Survey of Methodological Issues. Technical Report, pp. 96–167, Dept. of Software Eng. and Computer Science, University of Karlskrona/Ronneby (1996)
4. PASSI Toolkit, <http://sourceforge.net/projects/ptk>
5. Johnson, R., Foote, B.: Designing reusable classes. *J. Object-Oriented Programming* 1(2), 22–35 (1988)
6. Cubillos, C., Cabrera, D.: Towards an Agent Framework for a Passenger Transportation Virtual Enterprise. In: 4th International Conference on Web Information Systems and Technologies, Portugal, vol. 1, pp. 292–295 (2008)
7. Fontoura, M., Pree, W., Rumpe, B.: *The UML Profile Framework Architectures*. Addison Wesley, Reading (2000)

Interactive Learning of Expert Criteria for Rescue Simulations

Thanh-Quang Chu^{1,2}, Alain Boucher², Alexis Drogoul^{1,2}, Duc-An Vo^{1,2},
Hong-Phuong Nguyen³, and Jean-Daniel Zucker¹

¹ IRD, UR079-GEODES, 32 av. Henri Varagnat, 93143 Bondy Cedex, France

² AUF-IFI, MSI, ngo 42 Ta Quang Buu, Hai Ba Trung, Ha Noi, Viet Nam

³ IG-VAST, 18 Hoang Quoc Viet, Cau Giay, Hanoi, Vietnam

thanh.quang@gmail.com, alain.boucher@auf.org

alexis.drogoul@gmail.com, vdan@ififi.edu.vn

nhphuong@netnam.vn, jdzucker@gmail.com

Abstract. The goal of our work is to build a DSS (Decision Support System) to support resource allocation and planning for natural disaster emergencies in urban areas such as Hanoi in Vietnam. The first step has been to conceive a multi-agent environment that supports simulation of disasters, taking into account geospatial, temporal and rescue organizational information. The problem we address is the acquisition of situated expert knowledge that is used to organize rescue missions. We propose an approach based on participatory techniques, interactive learning and machine learning. This paper presents an algorithm that incrementally builds a model of the expert knowledge by online analysis of its interaction with the simulator's proposed scenario.

Keywords: Rescue Management, Multi-agent Simulation, Decision Support Systems, Knowledge Extraction, Participatory Learning, Interactive Learning.

1 Introduction

In cases of post-disaster situations in urban areas, there is a need for supporting human decision-making capacities with information technology tools. The first step in building a reliable decision-support system is to simulate the relief effort and to learn human strategies from various disaster scenarios. The devastated infrastructures and human casualties are input as GIS (Geographical Information System) data for the rescue simulation. Rescue teams, such as ambulances, firefighters or policemen are modeled and simulated by agents along with their behaviors. The DSS in this case must address two issues (Figure 1): the first is the ability to simulate different disaster scenarios integrating all available information, coming from either GIS data or other sources; the second is the ability to propose rescue solutions respecting the experts' criteria.

Concerning the first issue, i.e. building a simulator for rescue teams acting in large urban disasters, the Robocup-Rescue [13] community is proposing a multi-agent simulation environment for urban earthquakes. The goal of the agents (representing

firefighters, policemen and ambulance teams) consists in minimizing the damages caused by the earthquake. Damages include buried civilians, buildings on fire and blocked roads. The RoboCup-Rescue simulation environment is a useful test-bed for evaluating cooperative multi-agent systems. However, a major problem with RoboCup Rescue is its lack of support for standard GIS description, which prevents it from being compatible with existing GIS city descriptions. We have therefore developed a specific simulation platform that we will present briefly in Section 2.

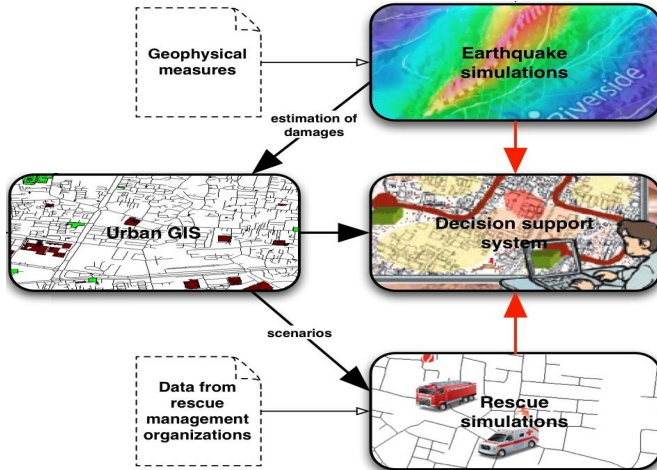


Fig. 1. The rescue simulation and the decision support systems are the results of solving the first and second problems respectively

This paper will be mainly devoted to the presentation of our solution regarding the second issue. In order to capture the experts' experience and knowledge, we propose using an original combination of participatory design and interactive learning. In this approach, the experts are invited to interact with the simulation by trying out different rescue scenarios and representing their decision criteria or modifying the agents' behaviors in order to improve the realism of the whole simulation and to optimize the relief activity.

This paper is organized as follows: Details of the first problem and its solution are presented in section 2 with an open rescue model based on the GAMA platform (GIS & Agent-based Modeling Architecture), an environment for implementing and running spatially explicit multi-agent simulations. In section 3, we present our solutions for the second problem with an example of ambulance rescue activities. Section 4 is our demonstration of the participatory design methodology and experimental protocol that has been used in the rescue simulation for our algorithm. Finally, we arrive at conclusions and discuss the experiments performed with various rescue scenarios for the ambulance team in section 5.

2 Open Rescue Model Based on the GAMA Platform

2.1 The GAMA Platform

The MSI team has developed the GAMA platform based on the Repast toolbox. It is a generic environment for experimentation that aims to provide experts, modelers and computer scientists the necessary tools for modeling and simulating spatially explicit multi-agent systems. The features of GAMA are: (1) the capacity for transparent use of complex data from GIS as an environment for the agents, (2) the ability to manage a large number of (heterogeneous) agents (3), the ability for automated and controlled experiments (by varying automatic settings, recording statistics, etc.), (4) the possibility for non-computer scientists to develop models or to interact with agents during the simulation execution; (5) the opportunity for modelers to easily extend available behavior models through the definition of a generic meta-model.

2.2 Open Rescue Model

We built a rescue simulation model on GAMA using urban GIS data from the Ba-Dinh district in Hanoi. This urban GIS is completed with estimations coming from an earthquake simulation model [8] (Figure 1). All GIS data is managed in the shapefile format (an open GIS standard) using the ArcGIS tool. This model provides building damage and casualty estimates in the observed areas.

Various levels of building damage, expressed as percentages, and casualty distribution is initialized by a semi-randomized localization method in the environment that will support the simulation agents in the rescue model.

As long as the environment is initialized, the most important part of this exercise is modeling the agents who will exhibit dynamic behavior in the course of the simulation.

Some agents come directly from the GIS environment. The program regards buildings, hospitals, emergency services etc. as agents that inherit corresponding attributes in the GIS (location, severity of injuries, damage level, etc.). This allows modelers to add certain dynamic behaviors such as the fire spreading through buildings, evolution in victims' health, evolution of building damage, the capacity of hospitals to treat injuries, the influence of building damage on occupants, and so on.

Other agents implemented in the model are victims and rescue teams: ambulances, firefighters or police officers (as in the Robocup-Rescue Simulation).

In the modeling process communication skills, autonomous behaviors, decision-making and the capacity to "remember" are all included.

Figure 2 is a view of a simple simulation including fires, victims, ambulances and firefighters. The firefighters' goal is to extinguish the fires and the ambulances' goal is rescuing the injured.

This simulation result is a good example of what should be expected from the first stage in decision support: situational awareness. The model gathers a good amount of (static, dynamic, geophysical, social, urban, etc.) information in a unique and user-friendly interface. This open rescue model allows the combination of GIS with multi-agent simulations. It establishes abstract models of the relief activities that can serve as a foundation for more realistic models.



Fig. 2. Open rescue model of the Ba-Dinh district in Hanoi showing fires, victims ambulances and firefighters

3 Interactive Learning for Rescue Simulation

3.1 Example of Rescue Decision with Ambulance's Behaviors

The ambulance's task is to take care of victims in its defined area. Their goal is to provide assistance to a maximum of injured victims thusly assuring as few deaths as possible. They must urgently perform on-the-fly first-aid, and/or transporting, with the least delay, injured victims to hospitals. An ambulance can normally carry several injured at the same time.

Figure 3 shows an example of an emergency situation. In this example, we assume that the ambulance A_1 (in the center of Figure 3) has access to all of the information inherent in the situation. If A_1 carries no victim, it must decide to which victim it will provide first-aid and to take the victim to the hospital if necessary. For example, A_1 may decide to go to the nearest victim V^5 , and then take him to hospital H^2 . But if A_1 knows that the victim V^6 is more seriously injured than V^5 , it could also decide to go first to V^6 . It is also possible that the victims V^1, V^2, V^3 are seriously injured and because they are all three near one-another, the ambulance A_1 may decide to go to this group, and then to take them to hospital H_2 , this choice will probably save three victims V^1, V^2, V^3 instead of one victim V^6 . Another possibility can happen: if the ambulance A_1 knows that the ambulance A_3 will take care of the group of victims V^1, V^2, V^3 and that the group of victims V^9, V^{10}, V^{11} are seriously injured, the ambulance A_1 may decide go to the group V^9, V^{10}, V^{11} and then take them to hospital H^3 . In this case, the solution will probably save six victims V^1, V^2, V^3 and V^9, V^{10}, V^{11} instead of three victims V^1, V^2, V^3 .

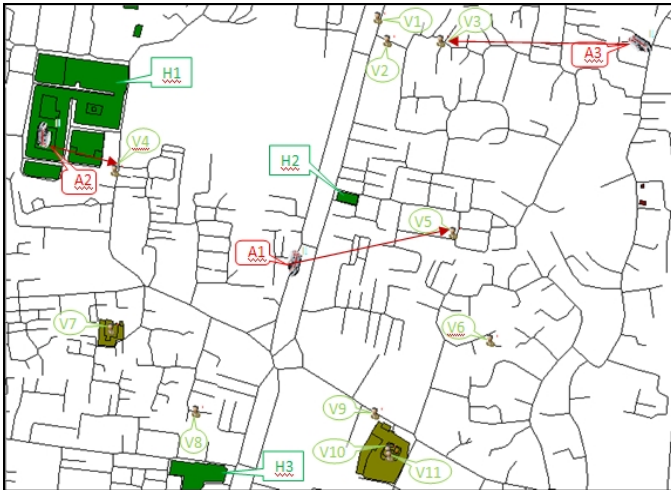


Fig. 3. Example of a possible emergency situation. A_x represents ambulances, V^x represents victims awaiting rescue and H^x are hospitals in the neighborhood. The red arrows link the ambulance to its best objective. The user can change any arrow to control the objective of each ambulance through this interactive interface.

3.2 Decision Criteria for Ambulance

The ambulance's decision may depend on a lot of information, so decisions must follow certain strategies to improve their relief activities. For example, after experiencing the above situations, the ambulance may acquire action rules as follows: the ambulance will first look at the "hot" positions on the map, where there are more victims who require assistance. The ambulance must take the victims' injury severity level into account: the more seriously injured victims should be tended to earlier. The ambulance should not go to places where other ambulances will eventually arrive, and so on. All of these rules are saved in the form of decision strategies for ambulances to be used in later situations.

Normally an ambulance does not have access to all the information related to its situation. A possible conflict can thus occur. Let us consider a situation where the ambulance is in the process of taking one or more seriously injured victims to the nearest hospital but is informed (by the information centre or the other ambulance via a communication channel) that a group of victims in need of rescue was spotted right along the road he is travelling. The decision is difficult: either go to the hospital, or go to the group of outside victims to practice first-aid. Several criteria might be involved in this decision, as shown in Table 1.

We have two types of criteria: criteria to maximize (+) which show that the victim or the hospital having greater values for these criteria will have higher priority in the ambulance's decision process; and criteria to minimize (-) which show that the victim or the hospital having lesser values for these criteria will have higher priority in the ambulance's decision process. Concerning the criteria of ambulance, for example, the autonomy (A) has the type of (V+)(H-) designating that the victims have higher priority in the situations when the ambulance has bigger value of autonomy, and the hospitals have priority if its autonomy is small.

Table 1. Decision criteria of the ambulance. The minus sign (-) indicates a criteria to minimize while the plus (+) sign indicates a criteria to maximize.

Attributes of	Criteria formula	Criteria type	Name
Ambulance A	Autonomy(A)	(V+)(H-)	C1
	Min-time-to-death(A)	(V+)(H-)	C2
Victim V	Time(A, V)	(-)	C3
	Time-to-death(V)	(-)	C4
	Time-to-nearest-other-victim(V)	(-)	C5
	Time-to-nearest-other-available-ambulance(V)	(+)	C6
	Time-to-nearest-available-hospital(V)	(-)	C7
Hospital H	Time(A, H)	(-)	C3'
	Time-to-nearest-other-victim(H)	(-)	C5'
	Time-to-nearest-other-available-ambulance(H)	(+)	C6'

3.3 Expert's Intervention in the Simulation Via Interactive Interface

Making appropriate decisions for the ambulances raises difficulties for two reasons. Firstly, there are too many criteria involved in this decision. Secondly, each ambulance only has partial knowledge about the situation (local view) at the moment of decision-making; thus, the ambulances lack both necessary information and the strategies to take good decisions. So, ambulances require user-intervention to make appropriate decisions by providing supplementary information about the situation. The user may provide their useful experience to aid in the decision, and/or they may force the ambulance to take a particular decision with or without explanation.

We assume that ambulance agents are constrained by an initial decision strategy. As long as the user considers the decisions taken by the ambulances optimal, s/he need not interact. On the other hand, if the user identifies a "better" solution, s/he may interact, by specifying or not upon which criteria he is basing his interaction. Thus, the user can "play the role" of the ambulance by forcing a decision (to continue on to the hospital or to stop and treat roadside victims). The objective is for the ambulance agents to gradually acquire, by collecting enough of these couples "information to combine—behavior to perform" and generalizing these cases (through adapted machine learning algorithms), a decision strategy that can be reused independently in close or similar circumstances.

Implementation of the interface between ambulances and users is necessary to ensure that ambulances can acquire expert knowledge during the learning process. This provides an excellent support for both situational awareness and user action during the course of the simulation.

For example, in figure 3, at any time, the objectives of the ambulance can be either victims or hospitals. The red arrow links the ambulance to its best objective. The user can pause the simulation and change the arrow to modify the objective of each ambulance.

Many of the lessons learned will serve for the implementation of an actual interface that can be used throughout the experiments with experts. The bottom line, not very surprising in this context, is that the interface should adapt, as much as possible, to the rhythm and needs of the user: giving him or her the possibility to change the speed of the simulation, to zoom in and out on situations, to dynamically change the colors and

shapes of the information displayed, to hide or reveal any pieces of information, to come back in time, etc. This appears to be a cognitive (and not simply cosmetic) necessity for the user to gain familiarity with the tool.

However, if we want to capture the knowledge mobilized *in situation*, we also need the interface to be as pressing and demanding as a critical real-world context could be imagined. Once the user is adept at manipulating the simulator, all experiments should then take place in *real-time*. Since we cannot, of course, ask an expert to play his/her role for 12 or 24 hours (as in reality), the learning sessions will be cut into time-bounded, incremental episodes with their own goal, learning task, and time limit, which will be presented later in the section 4.

3.4 Interactive Learning of Expert Criteria for Ambulance Decision

The idea behind the algorithm is the incremental construction of a utility function through expert intervention. The general form of the utility function that we have decided to consider is a weighted linear combination of criteria represented as follows:

$$F(V^k) = \sum w_i * C_i^k \quad (1)$$

Where V^k is the k^{th} victim; w_i : the weight of the i^{th} criteria and C_i^k : the value of the i^{th} criteria for the k^{th} victim

The victim V^{\min} will be selected if:

$$F(V^{\min}) = \text{Min}_k\{F(V^k)\} = \text{Min}(\sum w_i * C_i^k) \quad (2)$$

$$V^{\min} = \text{ArgMin}\{F(V^k)\} \quad (3)$$

To calculate the minimum value of the utility function, the value signs of the criteria to maximize are reversed. This is the case for the following criteria: *Time-to-nearest-other-available-ambulance(V)*, *Time-to-nearest-other-available-ambulance(H)*.

From a simple optimization function containing only the criterion $C_1^k = \text{Time}(A,V)$ with $w_1=1$, we have :

$$F(V^k) = C_1^k \quad (4)$$

As long as the user does not interact, the weights of the criteria will not be changed. The algorithm will remember all situations with the corresponding decision. If the user interacts and changes the objective of ambulance to V^{\min} , the algorithm will adapt the criteria weight to reflect this change: either by changing the weight of criteria in the function, either by adding new criteria weighted C_j in the function if no change can satisfy (3). We have the updated function:

$$F(V^k) = (\sum w_i * C_i^k) + w_j * C_j^k \quad (5)$$

The selection of the criteria C_j and the weight w_j are based on the algorithm calculating hyper-volume indicator of Pareto approximation set [15].

3.5 Example of the Algorithm’s Execution

To illustrate the execution of the proposed algorithm, let us come back to the situation described in figure 3. The problem is here to identify a rescue scenario for ambulance A_1 . A first set of criteria has been selected to run our preliminary experiments: see table 1. The detailed values for each criterion are given in table 2 hereafter.

Table 2. Values of the selected criteria related to both victims and hospitals in the context of the example described on Figure 3

Ambulance’ criteria			Victims’ criteria				
C1(V+)	C2(V+)		C3(-)	C4(-)	C5(-)	C6(+)	C7(-)
0	∞	V^1	6	14	To $V^2 = 1$	To $A_3 = 5$	To $H^2 = 4$
0	∞	V^2	6	14	To $V^1 = 1$	To $A_3 = 4$	To $H^2 = 4$
0	∞	V^3	7	14	To $V^2 = 1$	To $A_3 = 3$	To $H^2 = 5$
0	∞	V^4	10	12	To $V^7 = 4$	To $A_2 = 1$	To $H^1 = 1$
0	∞	V^5	4	18	To $V^1 = 6$	To $A_3 = 10$	To $H^2 = 2$
0	∞	V^6	12	18	To $V^{10} = 3$	To $A_2 = 14$	To $H^3 = 9$
0	∞	V^7	8	20	To $V^4, V^8 = 4$	To $A_2 = 5$	To $H^1, H^3 = 5$
0	∞	V^8	7	20	To $V^7 = 4$	To $A_2 = 7$	To $H^3 = 3$
0	∞	V^9	9	20	To $V^{10} = 0$	To $A_2 = 12$	To $H^3 = 6$
0	∞	V^{10}	9	20	To $V^{11} = 0$	To $A_2 = 12$	To $H^3 = 6$
0	∞	V^{11}	9	20	To $V^{10} = 0$	To $A_2 = 12$	To $H^3 = 6$
(H-)			Hospitals’ criteria				
(H-)	(H-)		C3’(-)		C5’(-)	C6’(+)	
0	∞	H^1	10	∞	1	To $A_2 = 0$	∞
0	∞	H^2	2	∞	2	To $A_3 = 8$	∞
0	∞	H^3	5	∞	3	To $A_2 = 8$	∞

At the beginning, the utility function F_i (where I represents the i th iteration of the utility function adjustment) of the ambulance A_1 is initialized using the following function:

$$F_1(V^k) = C_3^k \tag{6}$$

In other words, the decision-making of an ambulance is based on the third criteria (C_3^k) which according to table 1 represents the moving time from current position to the victims. Following this utility function, the victim V^5 as the nearest victim to rescue is thus selected (this is indicated in the user interface). At this time, an expert may intervene to change the decision of the ambulance’s decision, forcing its goal to switch to V^1 instead. Indeed, if the ambulance A_1 decides to rescue the victim V^1 instead of V^5 . It may, afterwards, have more advantage to rescue the victim V^2 , which lies very close to V^1 . Let us suppose that this is the reason why the expert made such modification on the fly.

At this point, based on the algorithm, and because the current string of criteria do not account for the choice of the expert, the utility function will be recomputed to accommodate for the expert modification and an additional criteria and an associate weight will be added. The algorithm suggests the parameter C_5 and an associated weight of 3 such that V^1 becomes the primary objective: $F(V^1) = Min\{F(V^k)\}$, and that

the previous decisions made by the ambulance before the intervention of the expert remain correct in light of the new utility function. The utility function is therefore updated to become:

$$F_2(V^k) = C_3^k + 3 * C_5^k \quad (7)$$

It basically states that, besides the criterion related to the *time* to reach the victim, the one called “*time-to-nearest-other-victim*” (computed for each victim) should now be taken into account as well in the decision of the ambulance. Moreover a more important weight is given to parameter C_5^k so as to guarantee that the function is consistent where previous situation.

We can see here that this procedure allows the utility function to incrementally improve, and the ambulance decision will more and more reflect the experts’ choice. If the function cannot be modified to accommodate for the expert’s decision it might be either because the expert uses more information than the agents, or the expert is not consistent, or that its decision cannot be captured by the function chosen. All different outcomes may happen in our experiments.

4 Experimental Protocol and Preliminary Results

As stated above, if we want to use rescue simulations as supports in decision-making processes, need to make them as realistic as possible, and to reflect, as much as possible, the decisions that would be taken by an expert in real situations. Most of these decisions are a compromise between existing regulations, written rules and, perhaps more importantly, the experience of the expert in similar situations.

Designing a process by which this latter aspect can be incorporated in the behavior of the agents with the help of learning techniques, will not only help in building more realistic simulations, but also increase the confidence of the expert in the support eventually provided by the simulation.

However, in order for this process to be successful, it has to be designed very carefully. [16] has proposed, in a similar context, to apply methodological advices derived from those employed in practices like *participatory design* or *user-centered design* in order to control the experimental process by which agents can acquire knowledge from the experts. Basically, and similarly to [16], our experiments will then rely on three major components: (1) The design of a flexible and ergonomic user-interface that would allow for real-time interactions between the expert and agents in the simulator as we mentioned above in the section 3.3. (2) The design of an experimental protocol composed of sessions organized around a goal, a set of learning tasks and a set of support scenarios. (3) The design of well-thought-out scenarios based on realistic conditions and corresponding to specific learning tasks and objectives.

Even when the actors are already at ease with the user interface of the simulator, the quality of the knowledge that might be captured will strongly depend on: (a) The commitment and motivation of the user (which is known to decrease over time); (b) The realism of the scenarios provided and their understanding by the user; (c) The focusing (in term of task, or goal) of the sessions during which the users will play their role.

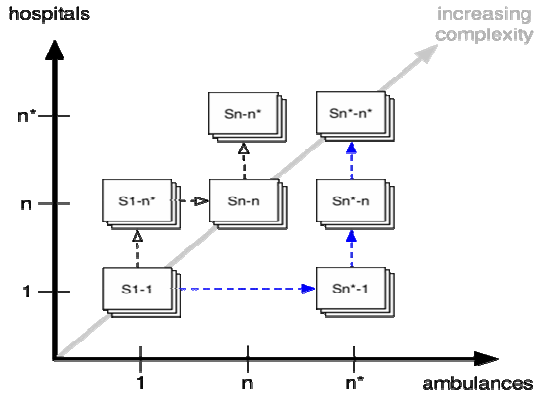


Fig. 4. Description of multiple scenarios as incremental representations of informational contexts of increasing complexity. In the bottom-left corner, the context only implies one ambulance and one hospital and therefore little information on which to base a decision. In the top-right corner, n^* indicates n agents able to communicate (for sharing information or coordinating their task), which represents the most complex situation agents can face if we only take hospitals and ambulances into account.

Therefore, a complete learning session will be organized as a succession of “episodes”, each of them being structured in the following way: (1) The task to be fulfilled by the agents and the timeframe within which they can accomplish it (for instance, save a maximum of victims in the minimum of time, save the most critical victims and communicate about the others, etc.) is communicated to the user and we make sure it is perfectly understood. Some episodes will of course share the same task. (2) For each task, a sequence of scenarios (see below) is then chosen, ranging from simple to complex. Each scenario will serve as a support for an “episode” of the session, and its results (in terms of machine learning) reused for the next episode in the sequence. (3) The set of criteria susceptible to be learnt (or ranked) during an episode depends on the complexity provided by the scenario. For instance, in basic scenarios, it may only contain the geographical location of the agents, while more advanced ones might want to take their communication (information received, etc.) into account.

There are many ways into which short-term focused scenarios could have been designed. Yet, we wanted a method that would allow for the learning episodes to act as different “layers” of increasing complexity, each of them focusing on the ranking of its own set of criteria and using the previous ones as starting points. As the criteria represents bits of information perceived, collected or received by the agents, we chose to base the progression of the scenarios on that of the “informational context” that the agents (and, therefore, the user) are facing. For instance, for a task like “locating and carrying a maximum of victims”, in a situation where only one ambulance and one hospital are being simulated (see figure 4), the decision of the agent will be based on a subset of the criteria used in a situation where several ambulances (or hospitals, or both) are present. And the criteria used in the latter situation will be themselves a subset of those necessary to take into account if all these agents are communicating or coordinating themselves.

Of course, the scenario's space can grow as needed to account for other agents (firemen, civilians, victims themselves, etc.) or criteria (communication of orders, changes in priorities, etc.). But we have to keep in mind that (1) not all of them are realistic; (2) no expert will be able to play them all. The path they will eventually follow, in their session, from one episode to the other, will be different from one expert to the other, and decided after each run through an interview with the modelers and an evaluation of their previous interactions with the agents.

5 Conclusions

Natural disaster management in urban areas is an extremely complex problem. In this context, we are interested particularly in the use of information technology, GIS and multi-agent simulation tools to address the vital problem of resource allocation for disaster response activities. Our research is intended to provide a means for building efficient decision support systems that would be easily usable by non-computer scientists.

There are several research projects (e.g. RoboCup-Rescue Simulation) which attempt to address similar questions relying on multi-agent models to optimize the planning of rescue teams, the combination of partial GIS and planning methods, and so on. However, there are few works that take into account the human (and subjective) aspects of decisions made during the disaster response.

In proposing a method enabling experts to interact directly with the agents of the simulation to teach them "good" behavior, we hope to (1) improve the realism of these simulations (and thus improve the strategies that can be learnt/proposed), (2) increase the confidence of decision-makers in the supporting decision tools; (3) facilitate training of the same decision-makers to these tools.

References

1. Özdamar, L., Yi, W.: Greedy Neighborhood Search for Disaster Relief and Evacuation Logistics. IEEE Computer Society, Los Alamitos (2008)
2. Paquet, S., Bernier, N., Chaib-draa, B.: An Online POMDP Algorithm for Complex Multi-agent Environments. In: AAMAS 2005, Utrecht, Netherlands (July 2005)
3. Takahashi, T.: Agent-Based Disaster Simulation Evaluation and its Probability Model Interpretation. In: ISCRAM 2007 (2007)
4. Suárez, S., López, B., de La Rosa, J.L.: Co-operation strategies for strengthening civil agents' lives in the RoboCup-Rescue simulator scenario. In: First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster, Workshop Padova (2003)
5. Farinelli, A., Grisetti, G., Iocchi, L., Lo Cascio, S., Nardi, D.: Using the RoboCup-Rescue Simulator in an Italian Earthquake Scenario. In: The program Agenzia 2000 of the Italian Consiglio Nazionale delle Ricerche
6. Paquet, S., Bernier, N., Chaib-draa, B.: Comparison of Different Coordination Strategies for the RoboCupRescue Simulation. In: Orchard, B., Yang, C., Ali, M. (eds.) IEA/AIE 2004. LNCS, vol. 3029. Springer, Heidelberg (2004)

7. Drogoul, A., Ferber, J.: Multi-Agent Simulation as a Tool for Modeling Societies: Application to Social Differentiation in Ant Colonies. In: Castelfranchi, C., Werner, E. (eds.) MAAMAW 1992. LNCS, vol. 830, pp. 2–23. Springer, Heidelberg (1994)
8. Nguyen-Hong, P.: Decision support systems applied to earthquake and tsunami risk assessment and loss mitigation. In: Proceedings of IHOCE 2005, Kuala Lumpur, Malaysia (2005)
9. Sempé, F., Nguyen-Duc, M., Boucher, A., Drogoul, A.: An artificial maieutic approach for eliciting experts' knowledge in multi-agent simulations. In: Sichman, J.S., Antunes, L. (eds.) MABS 2005. LNCS, vol. 3891. Springer, Heidelberg (2006)
10. Ramesh, R., Eisenberg, J., Schmitt, T.: Improving Disaster Management: The Role of IT in Mitigation, Preparedness, Response, and Recovery. Committee on Using Information Technology to Enhance Disaster Management, National Research Council, Washington, USA (2005)
11. Nguyen-Duc, M.: Vers la conception participative de simulations sociales: Application à la gestion du trafic aérien. Thèse de doctorat de l'université de Paris 6 (2007)
12. McCallum, A.K.: Reinforcement Learning with Selective Perception and Hidden State. PhD thesis, University of Rochester, Rochester, New-York (1996)
13. Paquet, S.: Distributed Decision-Making and Task Coordination in Dynamic, Uncertain and Real-Time Multiagent Environments. PhD thesis, Faculté de Sciences et Génie, Université Laval, Québec (2006)
14. Amouroux, E., Chu, T., Boucher, A., Drogoul, A.: GAMA: an environment for implementing and running spatially explicit multi-agent simulations. In: 10th Pacific Rim International Workshop on Multi-Agents (PRIMA), Bangkok, Thailand (2007)
15. Zinflou, A.: Système interactif d'aide à la décision basé sur des algorithmes génétiques pour l'optimisation multi-objectifs, Master thesis, Université du Québec à Chicoutimi, pp. 46–50 (2004)
16. Nguyen-Duc, M., Drogoul, A.: Using Computational Agents to Design Participatory Social Simulations. *Journal of Artificial Societies and Social Simulation* 10(45) (2007)

Modularity in Agent Programming Languages

An Illustration in Extended 2APL

Mehdi Dastani, Christian P. Mol, and Bas R. Steunebrink

Utrecht University
The Netherlands
{mehdi, christian, bass}@cs.uu.nl

Abstract. This paper discusses a module-based vision for designing BDI-based multi-agent programming languages. The introduced concept of modules is generic and facilitates the implementation of different agent concepts such as agent roles and agent profiles, and enables common programming techniques such as encapsulation and information hiding for BDI-based agents. This vision is applied to 2APL, which is an existing BDI-based agent programming language. Specific programming constructs are added to 2APL to allow the implementation of modules. The syntax and intuitive meaning of these programming constructs are provided as well as the operational semantics of one of the programming constructs. Some informal properties of the programming constructs are discussed and it is explained how these modules can be used to implement agent roles, agent profiles, or the encapsulation of BDI concepts.

1 Introduction

Modularity is an essential principle in structured programming in general and in agent programming in particular. This paper focuses on the modularity principle applied to BDI-based agent programming languages. There have been many proposals for supporting modules in BDI-based programming languages, e.g., [1, 2, 6, 7]. In these proposals, modularization is considered as a mechanism to structure an individual agent's program in separate modules, each encapsulating cognitive components such as beliefs, goals, events, and plans that together can be used to handle specific situations. However, the ways the modules are used in these approaches are different.

For example, in Jack [2] and Jadex [1], modules (which are also called capabilities) are employed for information hiding and reusability by encapsulating cognitive components that implement a specific capability/functionality of the agent. In these approaches, the encapsulated components are used during an agent's execution to create events and to generate plans that handle the events. It should be noted that Jadex extends the notion of capability by providing an import/export mechanism to connect different capabilities. In other approaches [6, 7], modules are used to realize a specific policy or mechanism in order to control nondeterminism in agent execution. For example, in [6] modules are considered as the 'focus of execution', which can be used to disambiguate the application and execution of plans. In [7] a module is associated with a specific goal indicating which and how planning rules should be applied to achieve that specific goal.

In these approaches, decisions such as when and how modules should be used during an agent's execution are controlled by the agent's execution strategy, usually implemented in the agent's interpreter. An agent programmer can control the use of modules during an agent's execution in a limited way either in terms of the functionality of those components or through conditions assigned to the modules. For example, in Jack [2] and Jadex [1] the interpreter searches the modules in order to determine how an event can be processed. In [6, 7], belief or goal conditions are assigned to modules such that an agent's interpreter uses the modules when the respective conditions hold.

Like in other approaches, we consider a module as an encapsulation of cognitive components. However, the added value of our approach is that an agent programmer has more control in determining *how* and *when* modules are used. In contrast to the abovementioned approaches, we propose a set of generic programming constructs that can be used by an agent programmer to perform a variety of operations on modules. In this way, the proposed notion of module can be used to implement a variety of agent concepts such as agent role and agent profile. In fact, in our approach a module can be used as a mechanism to specify a role that can be enacted by an agent during its execution. We also explain how the proposed notion of modules can be used to implement agents that can represent and reason about other agents.

In order to illustrate our approach we explain in the next section an extension of the agent programming language 2APL with modules. The syntax and operational semantics of the module-based 2APL are presented in sections 3 and 4 respectively. In section 5, we discuss how the proposed notion of modules can be used to implement agent roles and agent profiles. Finally, in section 6 we conclude the paper and indicate some future research directions.

2 Extending 2APL with Modules

2APL is a multi-agent programming language that facilitates the implementation of BDI-based agents. The 'classical' (i.e. non-modular) version of this programming language is presented in [3, 4]. In this paper, we extend 2APL with modules. In this extension, a 2APL multi-agent program is specified in terms of a set of modules each having a unique name. Initially, a subset of these modules is identified as the specification of individual agents. The execution of a 2APL multi-agent program is therefore the instantiation and execution of this subset of modules. A 2APL module is an encapsulation of cognitive components including beliefs, goals, plans, action specifications, and different sets of rules that generate and repair plans when they are applied. A 2APL module can create, instantiate, and process other 2APL modules. This implies that a 2APL module can include (be specified by) other 2APL modules. There are several operations that a module instance can perform on another one. These operations can be implemented by means of 2APL programming constructs designed to operate on modules.

One of these operations is to *create* a module instance based on a declared module in the multi-agent program. One module instance can be created by another module instance or an agent that is initially created as an instance of a declared module. In such a case, the *creating module instance* (also called the *owner module instance*) will assign a unique name to the *created module instance*. The owner module instance is the

only module instance that can operate on the created module instance until the created module instance is released.

One module instance can create several instances of one and the same module, e.g., an instance m_i of a declared module M can create instances k_1, \dots, k_j of another declared module K . Also, two module instances can create two instances of one and the same module, e.g., instances m_i and n_j of declared modules M and N can create instances k_i and k_j of another declared module K . Finally, one and the same module instance can be used by two different module instances, e.g., an instance k_i of a declared K can be used by instances m_i and n_i of declared modules M and N , respectively. For this purpose, a special type of module, called a *singleton* module, is introduced. While the ownership of a singleton module instance can be changed through create and release operations performed by different module instances, the state of the singleton module instance is invariant with respect to these operations, i.e., the state of a singleton module instance is maintained after one module instance releases it and another one owns it again.

The owner of a module instance can *execute* it in two different ways. First, the owner can execute its owned module instance and wait until the execution of the owned instance stops. In order to indicate when the owned instance stops (such that the owner's execution can be resumed), a stopping condition is provided as the argument of the execution operation. This condition, which is specified in terms of the internals of the owned module instance, is evaluated by the overall multi-agent system interpreter. Second, an owner can execute its owned module instance in parallel to its own execution. The execution of the owned module instance stops either by means of a stop condition (evaluated on the internals of the owned module instance) or explicitly by means of a stop action performed by the owner. The execute operations can be used to implement 'focus of execution' and goal processing as discussed in [6] and [7], respectively.

Besides executing a module instance, the internals of a module instance can be accessed by its owner module instance. In particular, an owner instance can *test* and *update* the beliefs and goals of its owned module instance. In order to control the access to the internals of a module instance, two types of modules are introduced: *public* and *private*. A private module instance can only be executed by its owner and does not allow access to its internals. In contrast to private modules, the internals of a public module instance are accessible to its owner. These operations can be used to implement capabilities as discussed in [1, 2]. It is worth noticing that a multi-agent system is the (only) owner of all module instances that initially constitute the individual agents.

3 Syntax

This section presents the complete syntax of the 2APL programming language. As the syntax of the 2APL programming language without modules is presented elsewhere [3, 4], we here highlight the modifications and discuss only the module-related programming constructs. The 2APL syntax for the multi-agent issues is presented by means of a specification language. Using this specification language, one can 1) declare a set of modules, 2) assign external environments to the modules which are then allowed to access the assigned environments, and 3) specify the creation of individual agents as

```

<MAS_Prog> ::= "Modules :" <module>+
            "Agents  :" (<agentname> <moduleIdent> [<int>])+
<module>   ::= <moduleIdent> ".2ap1" [<environments>]
<agentname> ::= <ident>
<moduleIdent> ::= <ident>
<environments> ::= "@"<ident>+

```

Fig. 1. The EBNF syntax of 2APL multi-agent systems extended with modules

instances of some of the declared modules. The syntax of this specification language is presented in Figure 1 using the EBNF notation. In the following, we use $\langle ident \rangle$ to denote a string and $\langle int \rangle$ to denote an integer. A 2APL multi-agent program can thus indicate which modules could be created during the execution of the multi-agent program. This is done by the declaration of a list of module names preceded by the keyword `Modules` (how the declared modules are implemented will be explained in section 3.2). From the set of declared modules, some will initially be instantiated as individual agents that constitute the implemented multi-agent system. The list of the names of the agents that should be created together with their corresponding module names and the number of to be created agents (i.e., the number of module instances to be created) is preceded by the keyword `Agents`. For each agent, $\langle agentname \rangle$ is the name of the individual agent to be created, $\langle module \rangle$ is the name of the module specification that implements the agent when it is instantiated, and $\langle int \rangle$ is the number of agents that should be created. When the number of agents is $n > 1$, then n identical agents are created. The names of these agents are $\langle agentname \rangle$ extended with a unique number. Finally, $\langle environments \rangle$ is the list of environment names to which the module has access. Note that this programming language allows one to create a multi-agent system consisting of different numbers of different agents each having access to one or more environments.

3.1 A 2APL Example

Suppose we need to build a multi-agent system in which one single manager and two workers cooperate to collect gold items in a simple cellular environment called blockworld. The manager coordinates the activities of the two workers by asking them either to explore the blockworld environment to detect the gold items or to carry the detected gold items and store them. For this example, which can be implemented as the following 2APL program, the manager module (i.e., `manager.2ap1`) specifies the initial state of the manager agent with the name `m` (the implementation of the manager module is explained later on). The manager module, and thus the manager agent `m`, can access the database environment. Note that only one manager agent will be initialized and created (line 7). Moreover, the worker module (`worker.2ap1`) specifies the initial state of two worker agents. Note that the names of the worker agents in the implemented multi-agent system will be indexed with numbers 1 and 2, i.e., there will be two worker agents with names `w1` and `w2` (line 8). Finally, two additional modules are declared to implement the explorer and carrier functionalities (line 4, 5). As we will see later on, these functionalities will be used by the worker agents. Note that both functionalities can access the blockworld environment.

```

1  Modules: // example.mas
2  manager.2apl   @database
3  worker.2apl
4  explorer.2apl  @blockworld
5  carrier.2apl   @blockworld
6  Agents:
7  m manager
8  w worker 2

```

3.2 2APL Module Specification

A 2APL module, which is also used to create individual agents, is implemented by means of a specification language. The EBNF syntax of this specification language is illustrated in Figure 2. The gray parts of the syntax are not related to modules and are already presented in [3, 4]. In this specification, we use $\langle atom \rangle$ to denote a Prolog-like atomic formula starting with a lowercase letter, $\langle Atom \rangle$ to denote a Prolog-like atomic formula starting with a capital letter, $\langle ground_atom \rangle$ to denote a ground atom and $\langle Var \rangle$ to denote a string starting with a capital letter.

Although explaining the complete set of 2APL programming constructs is not the focus of this paper, we give a brief and general idea of the basic non-module constructs. 2APL provides programming constructs to implement a module in terms of beliefs, goals, action specifications, plans, and reasoning rules. An agent's beliefs is a set of Horn-clauses and represent information the agent believes about itself and its surrounding environments. An agent's goals is a set of conjunctive ground atoms, where each conjunct represents a situation the agent wants to realize. The programming language provides different types of actions such as belief update actions (to modify beliefs), belief and goal test actions (to query beliefs and goals), actions to adopt and drop goals, to send messages, and to change the state of external environments. Besides these programming constructs, 2APL provides constructs to implement three types of reasoning rules. The planning goal rules (PG-rules) can be used to generate plans based on the agent's beliefs and goals. The procedure call rules (PC-rules) can be used to generate plans for the received internal and external events including messages. Finally, the plan repair rules (PR-rules) can be used to repair a plan whose execution has failed.

The first module-related construct is the use of keywords `public/private` and `singleton`. The owner of a public module instance can both execute as well as access the internals of the owned public module instance¹. However, the owner of a private module instance can only execute the module instance and cannot access its internals.

The `create(mod-name, mod-ident)` action can be used to create an instance of the module with the name `mod-name`. The name that is assigned to the created module instance is given by the second argument `mod-ident`. The owner of the module instance can use this name to perform further operations on it. A module instance with identifier `m` can be released by its owner by means of the `release(m)` action. If the module is not a singleton, then its instance will be removed/lost. However, if the module is a singleton, then its instance will be maintained in the multi-agent system such that it can be owned by another module instance that creates it again. It is important to note that a singleton module can only have one instance at a time such that it can always be accessed by means of the module name `mod-name`. It is also important to note that the subsequent

¹ Note that the owner itself can be an instance of either a public or a private module.

```

<2APL_Module> ::= ("private" | "public") [{"singleton"}
  ("Include:" <ident>
  | "BeliefUpdates:" <BelUpSpec>
  | "Beliefs:" <belief>
  | "Goals:" <goals>
  | "Plans:" <plans>
  | "PG-rules:" <pgrules>
  | "PC-rules:" <pcrules>
  | "PR-rules:" <prrules>)*
<BelUpSpec> ::= ( "{" <belquery> "}" <beliefupdate> "{" < literals> "}" )+
<belief> ::= ( <ground_atom> " " | <atom> " " : -" < literals> " " )+
<goals> ::= <goal> ( " , " <goal> )*
<goal> ::= <ground_atom> ( "and" <ground_atom> )*
<baction> ::= "skip" | <beliefupdate> | <sendaction> | <externalaction>
  | <abstractaction> | <test> | <adoptgoal> | <dropgoal>
  | <createaction> | <releaseaction> | <return> | <moduleaction>
<createaction> ::= "create(" <ident> " , " <ident> ")"
<releaseaction> ::= "release(" <ident> ")"
<return> ::= "return"
<moduleaction> ::= <ident> " " <maction>
<maction> ::= "execute(" <test> ")" | "executeasync(" [{"<test>}] ")"
  | "stop" | <test> | <adoptgoal> | <dropgoal> | <updBB>
<updBB> ::= "updateBB(" < literals> ")"
<plans> ::= <plan> ( " , " <plan> )*
<plan> ::= <baction> | <sequenceplan> | <ifplan> | <whileplan> | <atomicplan>
  | <miplan> | <mwhileplan>
<beliefupdate> ::= <Atom>
<sendaction> ::= "send(" <iv> " , " <iv> " , " <atom> ")"
  | "send(" <iv> " , " <iv> " , " <iv> " , " <iv> " , " <atom> ")"
<externalaction> ::= "@ " <iv> "(" <atom> " , " <Var> ")"
<abstractaction> ::= <atom>
<test> ::= "B(" <belquery> ")" | "G(" <goalquery> ")" | <test> "&" <test>
<adoptgoal> ::= "adopta(" <goalvar> ")" | "adoptz(" <goalvar> ")"
<dropgoal> ::= "dropgoal(" <goalvar> ")" | "dropsubgoals(" <goalvar> ")"
  | "dropsupergoals(" <goalvar> ")"
<sequenceplan> ::= <plan> " , " <plan>
<ifplan> ::= "if" <test> "then" <scopeplan> [{"else" <scopeplan>}]
<whileplan> ::= "while" <test> "do" <scopeplan>
<atomicplan> ::= "[ " <plan> "]"
<scopeplan> ::= "{" <plan> "}"
<pgrules> ::= <pgrule>+
<pgrule> ::= [<goalquery>] "< -" <belquery> "|" <plan>
<pcrules> ::= <pcrule>+
<pcrule> ::= <atom> "< -" <belquery> "|" <plan>
<prrules> ::= <prrule>+
<prrule> ::= <planvar> "< -" <belquery> "|" <planvar>
<goalvar> ::= <atom> "("and" <atom>)*
<planvar> ::= <plan> | <Var> | "if" <test> "then" <scopeplanvar> [{"else" <scopeplanvar>}]
  | "while" <test> "do" <scopeplanvar> | <planvar> " , " <planvar>
<miplan> ::= "if" <ident> " " <test> "then" <scopeplan> [{"else" <scopeplan>}]
<mwhileplan> ::= "while" <ident> " " <test> "do" <scopeplan>
<scopeplanvar> ::= "[ " <planvar> " ]"
<literals> ::= <literal> ( " , " <literal> )*
<literal> ::= <atom> | "not" <atom>
<ground_literal> ::= <ground_atom> | "not" <ground_atom>
<belquery> ::= "true" | <belquery> "and" <belquery> | <belquery> "or" <belquery>
  | "(" <belquery> ")" | <literal>
<goalquery> ::= "true" | <goalquery> "and" <goalquery> | <goalquery> "or" <goalquery>
  | "(" <goalquery> ")" | <atom>
<iv> ::= <ident> | <Var>

```

Fig. 2. The EBNF syntax of 2APL extended with modules

creation of a singleton module by another module instance, which may be assigned a different name, will refer to the same instance as when it was released by its last owner.

When a public or private module m is created/instantiated, the created instance can be executed by its owner through the action $m.execute(\langle test \rangle)$ or $m.executeasync([\langle test \rangle])$. The execution of a module instance by means of an `execute` action, performed by an owner, has two effects: 1) it suspends the execution of the owner module instance, and 2) it starts the 2APL deliberation process based on the internals of the owned module instance. The execution of the owner module instance will be resumed as soon as the execution of the owned module instance is terminated. The termination of the owned module instance² is based on the mandatory test condition (i.e., the argument of the `execute` action), which is continuously evaluated by the overall multi-agent system interpreter. As soon as this condition holds, a stop event `stop!` is sent to the owned module instance. The module instance could then start a cleaning operation after which it should broadcast a return event. For this we introduce an action `return` that can be executed by an owned module instance after which its execution is terminated. The execution of this final action broadcasts an event `return!` that is received by the overall multi-agent system interpreter after which the execution of the owner module instance is resumed. The owner agent will be notified about the return event immediately after its execution if resumed. The return event can be used by the owner to, e.g., release the owned module instance.

The execution of a module instance by means of the `executeasync` action is identical to `execute` action, except that the owner does not have to wait until the execution of its owned module instance terminates. In fact, the owner continues with its own execution in parallel with the execution of the owned module instance. The execution of the module instance can be halted either by providing a test condition (i.e., the optional argument of the `executeasync` action) or by means of the `stop` action performed by the owner module instance. Like the `execute` action, the test will be evaluated at the multi-agent system level and based on the internals of the module instance. The `stop` action performed by the owning module instance will send the `stop!` event to the owned module instance.

The owner of a public module instance can access and update the internals of the instance. In particular, the owner can test whether certain beliefs and goals are entailed by the beliefs and goals of its owned public module instance m through action $m.B(\varphi) \ \& \ G(\psi)$. Also, the beliefs of a module instance m can be updated by means of action $m.updateBB(\varphi)$. A goal can be added to the goals of a module instance m by means of $m.adopta(\varphi)$ and $m.adoptz(\varphi)$ actions. Finally, the goals of a module instance m can be dropped by means of $m.dropgoal(\varphi)$, $m.dropsubgoals(\varphi)$ and $m.dropsupergoals(\varphi)$ actions. As explained in [3, 4], these actions can be used to drop from an agent's goals, respectively, all goals identical to φ , all goals that are a logical subgoal of φ , and all goals that have φ as a logical subgoal.

3.3 Example Revisited

Given our working example, the manager module can be implemented as follows:

² The owner cannot do this because its execution has been suspended.

```

9 Private // manager.2apl
10 BeliefUpdates:
11 { carryGold(A) } Ready(A) { not carryGold(A) }
12 { not carryGold(A) } Busy(A) { carryGold(A) }
13 Beliefs:
14 worker(w1).
15 worker(w2).
16 divided(L,L1,L2) :- append(L1,L2,L), evenlySized(L1,L2).
17 Goals:
18 haveGold()
19 Plans:
20 send(w1,request,play(exp))
21 PG-rules:
22 haveGold() <- worker(A) and not carryGold(A) |
23 { @database(findGoal(A),L); if B(not L=[]) then {send(A,request,play(car,L)); Busy(A)} }
24 PC-rules:
25 message(w1,inform,gold(L)) <- divided(L,L1,L2) |
26 { [ @database(addGold(L1,w1),_); @database(addGold(L2,w2),_) ] }
27 message(A,inform,done(L)) <- worker(A) | { @database(removeGold(L,A),_); Ready(A) }

```

As illustrated, the goal of the manager *m* is to have gold items (line 18). Moreover, it has one initial plan through which it sends a request to worker *w1* to explore the blockworld environment (line 20). The first PC-rule of the manager agent indicates that when it receives a list of detected gold items (i.e., *gold(L)*) from worker *w1*, then it divides the received list into two evenly sized lists of gold items and stores them in its database (line 25, 26) (the manager agent could also add these lists to its beliefs, but the aim of the example is to show the use of different environments). The second PC-rule indicates that when a worker informs the manager that it has collected and carried its assigned gold items to a safe depot, then the manager removes the goal items from its database and updates its beliefs with the fact that the worker is ready to carry new gold items (line 27). In order to achieve its goal, the manager agent checks its database continuously to see if it has information about gold items to be collected by one of the worker agents that is not carrying gold (note that the information about gold items should be received from the worker agent that initially was asked to explore the blockworld). If it can find such information (a non-empty list of gold items) in its database, then it will send a request to the corresponding agent asking to carry the gold items and store them safely in the blockworld. The manager agent will update its beliefs to contain information that the agent is busy carrying gold.

```

28 Private // worker.2apl
29 BeliefUpdates:
30 { true } GoldInf(X) { gold(X) }
31 Beliefs:
32 manager(m).
33 PC-rules:
34 message(A,request,play(exp)) <- manager(A) |
35 { create(explorer, myexp);
36   myexp.execute( B(gold(L)) );
37   send(A, inform, gold(L)); adminGold(L);
38   release(myexp) }
39 message(A,request,play(car,L)) <- manager(A) |
40 { create(carrier, mycar);
41   mycar.updateBB( gold(L) );
42   mycar.execute( B(done() or error()) );
43   if mycar.B(done()) then send(A, inform, done(L)) else @blockworld(reset(),_);
44   release(mycar) }
45 adminGold(L) <- true | { if B( L=[X|R] ) then { GoldInf(X); adminGold(R) } }

```

The worker agent is an agent that waits for requests to either explore the blockworld environment or carry the gold items and store them. When it receives a request to explore the blockworld environment from the manager (line 34), it creates an explorer module instance and executes it (line 35, 36). Note that the halting condition of this module instance is the belief that gold items are detected. When the execution of the module instance is halted, the worker agent sends the information about the detected gold items to the manager (line 37), updates its beliefs with the information about the detected gold items (this action illustrates the use of abstract action), and finally releases the explorer module instance (line 38). The third PC-rule (line 45) implements the execution of the abstract action `adminGold(L)` by going recursively through the list of gold items and adding each of them to its beliefs. Finally, the second PC-rule of the worker agent (line 39) is responsible for carrying gold items by creating a carrier module instance (line 40), adding the gold item information to its beliefs (line 41), and executing it until either it has found the gold items (`done()` condition) or an error has occurred (`error()` condition).

```

46 Public // explorer.2apl
47 BeliefUpdates:
48   { gold(L) } Finish()   { not gold(L) }
49   { true }   Detected(L) { gold(L) }
50 Beliefs:
51   foundGold() :- gold(_).
52 Goals:
53   foundGold()
54 PG-rules:
55   foundGold() <- true | { @blockworld(sensegold(),L); Detected(L) }
56 PC-rules:
57   event(stop) <- true | { Finish(); return }

```

The explorer module, which is a public module, has the goal to find gold items (line 51). In order to achieve this goal, it performs a sense gold action in the blockworld and adds the information about the detected gold items (i.e., `gold(L)`) to its beliefs (line 55). Note that this belief information is the halting condition of the module instance. In this example, the final PC-rule (line 57) is to react to the stop event that is broadcasted by the platform when the explorer's stopping condition holds. The reception of this event causes a clean-up operation to be performed by deleting all information about gold items from its beliefs and performing a return action. This return action causes the execution to be handed back to the worker module. Note that the goal `foundGold()` is achieved as soon as `gold(L)` is added to its beliefs.

```

58 Public // carrier.2apl
59 BeliefUpdates:
60   { gold([X|R]) } Remove(X)   { not gold([X|R]), gold(R) }
61   { gold(X) }   Finish()     { not gold(X) }
62   { true }      Done()       { done() }
63   { true }      Error()      { error() }
64 Goals:
65   goldStored()
66 PG-rules:
67   goldStored() <- gold([X|R]) |
68     { @blockworld(pickUpGold(X,_); @blockworld(storeGold(X,_); Remove(X) }
69   goldStored() <- gold([]) | { Done() }
70 PC-rules:
71   event(stop) <- true | { Finish() ; return }
72 PR-rules:
73   @blockworld(pickUpGold(E,_);X <- true | { Error() }

```

Finally, the carrier module (also a public module) has a goal to store a list of gold items safely. This goal can be achieved by picking one gold item from the list, store it in a blockworld depot, and remove that gold item from the list of stored gold items. Note the use of two PG-rules (lines 67 and 69) to handle empty and non-empty lists of gold items. Similar to the explorer module, the carrier module does a clean-up operation and performs the return action when it receives a stop event (line 71). The plan repair rule (line 73) adds error information (i.e., `error()`) to its beliefs when the execution of the `pickUpGold` action in the blockworld environment fails. Note that `error()` in the beliefs was one of the halting conditions to stop the execution of the carrier module instance. It is also important to note that it is up to the blockworld programmer to determine when the execution of the `pickUpGold` action fails.

4 Semantics

The semantics of 2APL is defined in terms of a transition system, which consists of a set of transition rules for deriving transitions. A transition specifies a single computation/execution step by indicating how one configuration can be transformed into another. In this paper, we first present the multi-agent system configuration, which consists of the configurations of individual agents and the state of the external shared environments. Then, due to space limitation, we present only one transition rule to illustrate how a multi-agent system transition (an execution step) can be derived. Here, we do neither present the configuration nor the transitions rules for individual agents. Elsewhere [4] we have presented the semantics of 2APL *without modules*. The execution of module-related programming constructs affect mainly the multi-agent system configuration. The only effect of the module-related actions at the individual agent level is that these actions are removed from the agent's plans upon execution. It is important to note that individual agent *transitions* are used as conditions of the multi-agent system *transition rules*.

The configuration of a multi-agent system is defined in terms of the configuration of modules instances (including agents) and the state of the external environments. The configuration of a module instance includes 1) an instance of the module (consisting of beliefs, goals, plans, events, and reasoning rules) with a unique name, 2) the name of the (parent) module that has created the module instance, 3) the identifier of the module specification,³ 4) a flag indicating whether the module instance is executing, and 5) the stopping condition for the module instance. Finally, the state of a shared environment is a set of facts that hold in that environment.

Definition 1 (multi-agent system configuration). *Let (A_i, p, r, e, φ) be a module configuration, where A_i is a module instance with the unique name i , p is the name of the owner of the module instance, r is an identifier referring to the module specification, e is the execution flag, and φ is the execution stopping condition. Let \mathcal{A} be a set of module configurations and χ be a set of external environments, each a consistent set of atoms (atom). The configuration of a 2APL multi-agents system is then defined as $\langle \mathcal{A}, \chi \rangle$.*

³ Note that there may be several instances of a module specification in a multi-agent system.

The initial configuration of a multi-agent system consists of the initial configuration of its individual agents and the initial state of the shared external environments as specified in the multi-agent program. The initial configuration of each individual agent is determined by the module that is assigned to the agent in the multi-agent program. The initial state of the shared external environment is set by the programmer, e.g., the programmer may initially place gold at specific positions in a blockworld environment.

In particular, for each individual agent implemented as $(i : m N)$ (which is preceded by the keyword `Agents :`) in the multi-agent program, N module/agent instances $(A_{i_1}, \text{mas}, m, \mathbf{t}, \perp), \dots, (A_{i_N}, \text{mas}, m, \mathbf{t}, \perp)$ are created and added to the set of module instances \mathcal{A} . Also, all environments that are assigned to a module in the multi-agent program are initialized and collected in the set χ . Note that all module instances that are created when the multi-agent program is initialized have `mas` as parent, `t` (true) as execution flag, and `⊥` as stopping condition.

The execution of a 2APL multi-agent program modifies its initial configuration by means of transitions that are derivable from transition rules. In fact, each transition rule indicates which execution step (i.e., transition) is possible from a given configuration. It should be noted that for a given configuration there may be several transition rules applicable. An interpreter is a deterministic choice of applying transition rules in a certain order.

Due to space limitation, we will present here only the transition rule for the creation of a non-singleton module. For the complete presentation of the formal semantics see [5].

In this transition rule, which is presented below, we use $A_i \xrightarrow{\alpha!} A'_i$ to indicate that the module instance configuration A_i can make a transition to module instance configuration A'_i when its execution results in the performance of action α (and thus *broadcasting* event $\alpha!$). Finally, we assume that *singleton*(r) holds if and only if the module r is a singleton module.

$$\frac{(A_i, p, r', \mathbf{t}, \varphi) \in \mathcal{A} \ \& \ A_i \xrightarrow{\text{create}(r,n)!} A'_i \ \& \ \neg \text{singleton}(r) \ \& \ \neg \exists r'', e, \varphi' : (A_{i,n}, i, r'', e, \varphi') \in \mathcal{A}}{\langle \mathcal{A}, \chi \rangle \longrightarrow \langle \mathcal{A}', \chi \rangle} \quad (1)$$

where $\mathcal{A}' = (\mathcal{A} \setminus \{(A_i, p, r', \mathbf{t}, \varphi)\}) \cup \{(A'_i, p, r', \mathbf{t}, \varphi), (A_{i,n}, i, r, \mathbf{f}, \perp)\}$.

The transition rule indicates the effect of the `create`(r, n) action performed by the execution of module instance A_i (the owner module instance), where r is the identifier of a non-singleton module specification (of which an instance should be created), and n is the name that will be assigned to the created module instance. This transition rule requires that the owner module instance i is in the execution mode (i.e., the execution flag equals `t`) and that there is no module instance with the same name already created by the same module (i.e., $\neg \exists r'', e, \varphi' : (A_{i,n}, i, r'', e, \varphi') \in \mathcal{A}$). The result is that the set of modules \mathcal{A} is modified and extended. In particular, the creating module instance is modified as it has performed the `create` action and the newly created module instance is added to the multi-agent system configuration. Note that the newly created module is not in execution mode (i.e., the execution flag equals `f`) and its stopping condition is set to `falsum` `⊥`. Note also that the stopping condition will be changed when the module is executed.

5 Roles, Profiles, and Task Encapsulation

The proposed module extension of 2APL is general enough to be useful for the implementation of several agent-oriented programming topics. These include the implementation of agent roles, agent profiles, and encapsulation of cognitive attitudes.

5.1 Agent Roles

The run-time creation and execution of a module instance can be used to implement the activation and enactment of a role. The module specification should then be considered as the specification of the role. In particular, the action `create(role, name)` can be seen as the activation of a role, by which the activating agent (owner) acquires a lock on the activated role, i.e., it becomes the role's owner and gains the exclusive right to manipulate the activated role. Note that when the *role* has been declared as *singleton*, this property of locking is important, because other agents cannot acquire the *role* as well. If *role* is not singleton, the role is created new and private to the creating agent anyway. Upon releasing a singleton role, the role is not deleted but retained with a blank owner, so that another agent may activate (using `create(role, name')`) and use it.

An agent that has successfully performed the action `create(role, name)` is the owner of *role* and may *enact* this role using `name.execute(φ)`, where φ is a stopping condition, i.e., a composition of belief and goal queries. The owner agent is then put on hold until the role satisfies the terminating condition, at which point control is returned to the owner agent. Alternatively, the role may be executed using `name.executeasync(φ)`, meaning that *role* will run parallel to the owner agent. Note that supplying as terminating condition $\varphi = \perp$ means that the role can only be stopped by executing `name.stop`, which of course is only possible if the role was enacted using `executeasync`. In principle, it is allowed for a role to activate and enact a new role, and repeat this without (theoretical) depth limits. However, this is usually not allowed in literature on roles. But it is up to the programmer to prevent roles from enacting other roles.

5.2 Agent Profiles

An agent can easily create and maintain profiles of other agents by creating non-singleton module instances. For example, assume agent *bas* executes the actions `create(profile_template, chris)` and `create(profile_template, mehdi)`, i.e., it uses a single template (specified as being *public*) to initialize profiles of the (hypothetical) agents *chris* and *mehdi*. These profiles can be updated by *bas* using e.g. `chris.updateBB(φ)` and `mehdi.adoptgoal(κ)` when appropriate. *bas* can even 'wonder' what *chris* would do in a certain situation by setting up that situation using belief and goal updates on *chris* and then performing `chris.execute(φ)` (or `executeasync`) with a suitable stopping condition φ . The resulting state of *chris* can be queried afterwards to determine what *chris* 'would have done'.

5.3 Task Encapsulation

Modules can also be used for the common programming techniques of encapsulation and information hiding. Modules can encapsulate certain tasks, which can be

performed by its owning agent if it performs an `execute` action on that module instance. Moreover, a module that has been declared to be *private* cannot be modified (e.g. by `updateBB`) by its owning agent. Such a module can thus hide its internal state and keep it consistent for its task(s). An important difference between *creating* a module (in the sense proposed here) and *including* a module (in the sense of [1, 2]) is that the contents of an *included* module instance are simply added to the including agent, whereas the contents of a *created* module instance are kept in a separate scope. So when using the `create` action, there can be no (inadvertent) clashes caused by equal names being used in different files for beliefs, goals, actions, and rules.

6 Conclusions and Future Work

In this paper we have introduced a mechanism to implement modules in BDI-based agent programming languages. We have illustrated this mechanism by extending the syntax and (operational) semantics of 2APL with transition rules for module-related actions that allow module instances to be created, executed, queried, modified, and to be released again. Each module instance is allowed to create other modules, and so on, up to a (theoretically) unlimited depth. Furthermore, by using the *public/private* and *singleton* flags in the specification of a module, the programmer can use these modules for common programming techniques such as data hiding and singleton access. We have also shown how modules can be used to facilitate the implementation of notions relevant to agent programming; namely, the implementation of agent roles and agent profiles. We intend to provide a proof of concept of the proposed extension by implementing the presented operational semantics in the current 2APL platform. It should be noted that modularity in programming languages is not new. Our proposed notion of modules is inspired on the concepts found in many languages, particularly object oriented languages. As a consequence some properties are the same, e.g. modules have an owner, which dictate the life cycle of the module. Also a module is designed with a particular task in mind, hiding the detail for the owner.

For future work, there are several extensions to this work on modularization that can make it more powerful for encapsulation and implementation of roles and agent profiles. First, the `execute` and `executeasync` actions may not be entirely appropriate for the implementation of profile execution, i.e., when an agent wonders “what would agent *X* (of which I have a profile) do in such and such a situation?”. This is because executing a profile should not have consequences for the environment and other agents, so a module representing an agent profile should not be allowed to execute external actions or send messages. Second, the notion of singleton can be generalized by introducing the possibility of specifying a minimum and maximum amount of instances of a module that can be active at one time. This can be used for ensuring that, e.g., there must always be three to five agents in the role of security guard. Third, new actions `add` and `remove` can be introduced that accept as arguments a module instance and a plan or rule, so that all types of contents of 2APL module instances can be modified during runtime. In particular, by creating an empty module instance and using `add` actions, modules instances can be created from scratch with custom components available at runtime.

References

1. Braubach, L., Pokahr, A., Lamersdorf, W.: Extending the Capability Concept for Flexible BDI Agent Modularization. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2005. LNCS, vol. 3862, pp. 139–155. Springer, Heidelberg (2006)
2. Busetta, P., Howden, N., Ronnquist, R., Hodgson, A.: Structuring BDI Agents in Functional Clusters. In: Jennings, N., Lesperance, Y. (eds.) ATAL 1999. LNCS, vol. 1757, pp. 277–289. Springer, Heidelberg (2000)
3. Dastani, M.: 2APL: A practical agent programming language. *International Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 16(3), 214–248 (2008)
4. Dastani, M., Meyer, J.-J.: A practical agent programming language. In: Dastani, M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS, vol. 4908. Springer, Heidelberg (2008)
5. Dastani, M., Mol, C.P., Steunebrink, B.R.: Modularity in Agent Programming Languages: An Illustration in Extended 2APL. Technical Report UU-CS-2008-022, Department of Information and Computing Sciences, Utrecht University (2008)
6. Hindriks, K.: Modules as policy-based intentions: Modular agent programming in goal. In: Dastani, M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS, vol. 4908. Springer, Heidelberg (2008)
7. van Riemsdijk, M.B., Dastani, M., Meyer, J.-J.C., de Boer, F.S.: Goal-Oriented Modularity in Agent Programming. In: Proceedings of AAMAS 2006, pp. 1271–1278 (2006)

On the Pheromone Update Rules of Ant Colony Optimization Approaches for the Job Shop Scheduling Problem

Dong Do Duc¹, Huy Q. Dinh^{2,3}, and Huan Hoang Xuan¹

¹ Department of Computer Science, College of Technology,
Vietnam National University, Hanoi, 144 Xuan Thuy, Hanoi, Vietnam

² Gregor Mendel Institute of Molecular Plant Biology, Vienna, Austria

³ Center for Integrative Bioinformatics Vienna, Vienna, Austria
dongdoduc@yahoo.com, huy.dinh@gmi.oeaw.ac.at, huanhx@vnu.edu.vn

Abstract. Ant Colony Optimization (ACO) system is an intelligent multi-agent system of the interacting artificial ants to solve the combinatorial optimization problems. Applying ACO approach in the typical NP-hard problem like job shop scheduling (JSS) problem is still an impressive and attractive challenge with the community. This paper proposes two improvements of ACO algorithm based on the convergence property of pheromone trails. Our improvements are better in both terms of accuracy and running time than the state-of-the-art Max-Min ant system by the simulation with the standard data sets.

Keywords: Ant colony optimization algorithm, job shop scheduling problem, ACO convergence.

1 Introduction and Related Work

Ant Colony Optimization (ACO), firstly introduced in [1] as an efficient algorithm for traveling salesman problem [2] as well as a meta-heuristic framework for combinatorial optimization problems [4], is one of the most recent approaches to solve the NP-hard problems motivating from the foraging mechanism of real ant colonies. Each ant leave a chemical substance (so-called *pheromone trail*) on the ground during finding the path from their nest to food source and vice versa. And the ants follow the path with pheromone trails and eventually concentrate on the shortest one. In ACO, they used the *artificial ants* as the intelligent agents together with the combination of the existed heuristic information of the solving problem and the reinforcement of ants' associated pheromone trails [8].

The interacting communication between the ants via the pheromone trails during the simultaneously search of shortest paths is the reason why ACO system performs as a multi-agent system. Almost all applications in ACO ([4] and references therein) have shared a pipeline: converting the considering problem to the shortest path problem over a given weighted graph, defining the corresponding pheromone trails and following the way in the first application to solve the

traveling salesman problem [2]. The ACO algorithms can be divided into three classes: Ant System(AS, [2]), Ant Colony System (ACS, [3]) and Max-Min Ant System (MMAS, [5]). Two later ones are the extensions of the first one with two different ways to update the pheromone intensities during the search for the shortest path of every ants.

Job shop scheduling (JSS) problem is one of the long-existed scheduling problems in the literature [11]. Studying this problem in the context of ACO started in [9] with the original Ant System algorithm. And the result, where the 6-job, 6-machine problem was investigated, is limited. JSS is a type of problem having poor heuristics for applying ACO, then we want to mainly investigate the pheromone trail behaviors, and control the suitable ACO parameters for this such special type of scheduling problem.

This paper gave the property of the pheromone trail convergences and proposed the new update rules for the job shop problem. Although the convergences of ACO algorithms have been investigated in [6], their results do not help much in real problem. The most recent paper [7] analyzed the convergence properties when the loop $t \rightarrow \infty$, and the result still did not show the insight of convergence behaviors that can help to control the pheromone trail updates. In this paper, we are motivated from the behavior of the convergence of pheromone trails on the edges not belonged to the best solution from a specific loop T . This such property suggested the improvements of the existed pheromone update rules for applying ACO in JSS problem and the simulation results confirmed the efficiency.

2 ACO Framework for JSS Problem

2.1 Description of JSS

This problem is denoted by a standard model $n/m/G/C_{max}$, where G indicates that jobs are connected to the technological production rules that describes their implementation order of machines. For example, a below T matrix $T = \begin{pmatrix} M_1 & M_2 & M_3 \\ M_2 & M_3 & M_1 \end{pmatrix}$ where a row represents the jobs and indicates the order of machines to be scheduled, each element of matrix T represents a specific operation. And the processing time of each operation can be represented as a following

matrix $P = \begin{pmatrix} t(O_{11}) & \dots & t(O_{1m}) \\ \dots & \dots & \dots \\ t(O_{21}) & \dots & t(O_{2m}) \\ \dots & \dots & \dots \\ t(O_{n1}) & \dots & t(O_{nm}) \end{pmatrix}$ where O_{ij} indicates that the operation j of

job i will be implemented by machine T_{ij} , and $t(O_{ij})$ is the corresponding implementation time. Then, a job shop problem can be represented as a pair (T, P) . The parameter C_{max} is minimum make-span of the job shop that considered as an objective function for the machine scheduling problem.

2.2 Solution Construction

[2] proposed an idea to define the n -job, m -machine JSS problem as a graph for ACO. Using the matrix T described above, a graph is constructed as followed:

the nodes represent the operations given in the T matrix, and the set of edges can be divided into two classes: the unidirectional horizontal edges, connected by the nodes of the same job, represent the technological order of processing a job and the bidirectional edges indicate other possibilities of an ant to make decision for choosing the next node in its path. Hence, the feasible schedule is the path from the starting node along the graph and visit all nodes, horizontally (the maximum number of nodes for a graph is $(n * m) + 1$). Afterward, a solution of each ant will be mapped onto a feasible machine schedule by implementing the operations sequentially. And the condition for implementing an operation is : the previous operations in the same job were done and the corresponding machine is free at that time. For illustration (see Fig.1), we give an example of

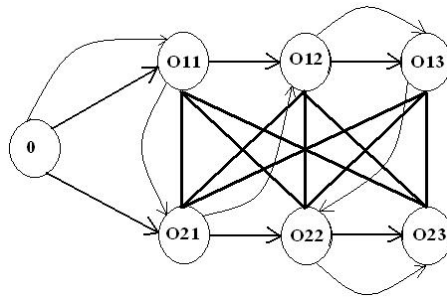


Fig. 1. An example of the $2/3/G/C_{max}$ job shop problem and one feasible solution built from ACO algorithms

the $2/3/G/C_{max}$ job shop so that we have 6 operations $O_{ij}, i = 1, 2; j = 1, 2, 3$ and the mentioned matrix T represents the machinery order for each job at each row. The artificial node O indicates the starting node of the ant. The unidirectional edges here described the order of the operations which need to be implemented (e.g O_{11} have to be implemented before the O_{12}), the bidirectional edges showed the other ability of choosing the next node for each ant from the current node like mentioned above. The curve arrows showed one path of the ant $O_{11} \rightarrow O_{21} \rightarrow O_{12} \rightarrow O_{13} \rightarrow O_{22} \rightarrow O_{23}$. We can see from that solution, the machines implement the operation O_{11}, O_{21} are machine 1 and 2, respectively. And the operation O_{12} is implemented by machine M_2 , after the O_{11} and O_{21} . This such graph structure was efficiently implemented with AS system in [9].

2.3 Pheromone Trail and Heuristic Information

Each edge with the corresponding pair (τ_{ij}, d_{ij}) representing its amount of pheromone trail and the heuristic distance between its nodes. Here, the very simple heuristic information - the processing time of the operation of node j can be used as a heuristic distance for edge (i, j) . The heuristic distance value can be easily looked up in the P matrix.

3 Convergence Property and Improvements of ACO

3.1 Convergence Property for Pheromone Trails

Pheromone update rules in ACO. Almost all the pheromone update rules in ACO based on two typical rules of ACS and MMAS system using the max-min rule for constraining the pheromone trails in both.

ACS-based rule. The rule consists of both local and global update rule. The pheromone trails are locally updated when an ant goes through the edges, following:

$$\tau_{u_i, u_{i+1}}(t + 1) \leftarrow (1 - \rho)\tau_{u_i, u_{i+1}}(t) + \rho\tau_1 \tag{1}$$

where $\tau_{u_i, u_{i+1}}$ indicates the value of pheromone trail associated on the edge (u_i, u_{i+1}) , $\rho < 1$ is a positive coefficient representing the so-called evaporation ratio according to a certain time, τ_1 , usually greater than the initial value τ_0 is a pre-defined positive constant. At the end of each iteration, the pheromone trails belonged to the best solution $w(t)$ and the others will be globally updated by following rule

$$\tau_{u_i, u_{i+1}}(t + 1) = \begin{cases} (1 - \rho)\tau_{u_i, u_{i+1}}(t) + \rho g(w(t)) & (u_i, u_{i+1}) \in w(t) \\ \tau_{u_i, u_{i+1}}(t) & \text{otherwise} \end{cases} \tag{2}$$

where $g(w(t))$ is the objective value of the solution $w(t)$

MMAS-based rule. After each ant completed building a solution, the pheromone trails are updated at each loop following :

$$\tau_{u_i, u_{i+1}}(t + 1) \leftarrow (1 - \rho)\tau_{u_i, u_{i+1}}(t) + \Delta_{u_i, u_{i+1}} \tag{3}$$

where

$$\Delta_{i,j} = \begin{cases} \rho g(w(t)) & (u_i, u_{i+1}) \in w(t) \\ \max\{\tau_1 - (1 - \rho)\tau_{u_i, u_{i+1}}(t), 0\} & \text{otherwise} \end{cases} \tag{4}$$

where $w(t)$ can be either the iteration-best solution or best-so-far solution, and $g(w(t))$ is the value of objective function of the solution $w(t)$.

Convergence property of pheromone trails. [7] used the stochastic process for investigating the convergence properties of τ_{ij} when $t \rightarrow \infty$. But it can not help much for application, then we showed that the behavior of pheromone trails will follow the below property from the loop T . Assume that an edge (u_i, u_{i+1}) belongs to and admissible solution and there exists T such that $(u_i, u_{i+1}) \notin w(t)$ for all $t \geq T$. We have the following:

- (i) In the ACS-based rule, $\tau_{u_i, u_{i+1}}(t)$ converges in probability to τ_1 ;
- (ii) In the MMAS-based rule, $\tau_{u_i, u_{i+1}}(t) = \tau_1$ for all t satisfying $t > T + \frac{\ln(\tau_1/g^*)}{\ln(1-\rho)}$ where $g^* = f(s)$ with $f(s)$ is the objective function value of an arbitrary solution belonged to previous iterations.

Based on the bounds in the above property, we proposed the new update rules following some quantitative insights into the pheromone behaviors. And they are especially verified with JSS problem by experimental results.

3.2 The Improvements for JSS Problem

Balance of intensification and diversification. For the type of problem which has lack of heuristic information like job shop problem, if the difference of τ_{min} and τ_{max} is large, the pheromone trails on “good” may decrease to τ_{min} so fast by chance. These edges should give the chance to be chosen by making τ_{min} and τ_{max} closer. Then we choose the ratio $1/2$ for τ_{min}/τ_{max} to increase diversification ability for the ACO algorithms. The reason why is that when we want to apply ACO to solve the JSS problem, the reinforced information from the pheromone trails is mainly focused on instead of the poor heuristic information. The search of ants will be the random search when this such ratio is approached to 1. With this such type of problem, we believe that the diversification and intensification is more balanced thanks to the suitable chosen τ_{min}/τ_{max} ratio and the scaling parameters α and β for the probability to choosing the next node of ants (see [2]).

Extension of MMAS-based rule. The so-called *smooth* MMAS system (SM-MAS) is proposed following the below update rule:

$$\tau_{u_i, u_{i+1}}(t + 1) = (1 - \rho)\tau_{u_i, u_{i+1}}(t) + \rho\Delta_{u_i, u_{i+1}} \tag{5}$$

where

$$\Delta_{u_i, u_{i+1}} = \begin{cases} \tau_{max} & (u_i, u_{i+1}) \in w(t) \\ \tau_{min} & \text{otherwise} \end{cases} \tag{6}$$

With the job shop problem, we realized that MMAS update rules will let the pheromone trail at the “bad” edges to τ_{min} fast although these edges may be potential for further solutions. We chose SMMAS with the pheromone update rule described above for letting the pheromone trail values of the such “bad” edges slowly decrease. And in the simulation, the running time is decreased because we do not need to control the pheromone trail values when they are not belonged to the interval $[\tau_{min}, \tau_{max}]$.

Extension of ACS-based rule. [3] proposed the ACS to improving the search performance by adding the local update rule. However, this approach does not escape the bad solution space during the search although the diversification is enhanced by setting up the interval $[\tau_{min}, \tau_{max}]$ for the pheromone intensity values. Then, multi-level ant system approach [8] (MLAS) is proposed to address by increasing τ_{mid} and τ_{max} for balancing the random search and the reinforcement from the pheromone trails. The MLAS algorithm can be considered as an extension of SMMAS in which we removing the evaporation process for the edges which is not visited by ants for a long time. The efficiency was proved by the experimental results with traveling salesman problem [8]. In job shop problem, we used the middle bound τ_{mid} is permanently kept as $1/2\tau_{max}$ for guaranteeing the suitable balance of intensification and diversification as mentioned above. We do not use the parameter τ_{min} because the pheromone trails on all edges are initialized by τ_{max} . Then, the parameters τ_{mid} and τ_{max} will control the the pheromone update as below:

- Online update : the pheromone trails belonged to the path of ants will be updated as follow

$$\tau_{u_i, u_{i+1}}(t+1) = (1 - \rho)\tau_{u_i, u_{i+1}}(t) + \rho\tau_{mid} \quad (7)$$

- Offline update : the pheromone trails belonged of the path of the best-so-far ants will be updated as follow

$$\tau_{u_i, u_{i+1}}(t+1) = (1 - \rho)\tau_{u_i, u_{i+1}}(t) + \rho\tau_{max} \quad (8)$$

Although the balance of the intensification and the diversification can be controlled by adjusting the τ_{mid}/τ_{max} ratio [8]. The experimental result (in the next section) will show the efficiency of this pheromone update rule although it is not easy to control and adjust the mentioned ratio.

Complexity. All of three considered algorithms implement nc loops, at each loop there are k ants simultaneously find the their own solution. And with the described graph, one ant will consequently visits $n * m$ nodes where n is the number of jobs and m is the number of machines in job-shop scheduling problem. Therefore, each algorithm need at least $O(nc * k * m * n)$. The MMAS algorithm needs $O(n * m)^2$ for the evaporation, and the same time for removing the edges whose pheromone intensity value is not belonged to the interval $[\tau_{min}, \tau_{max}]$. Hence, the complexity of MMAS system for job-shop scheduling is $O(nc * (k * m * n + (n * m)^2))$. Our smooth MMAS theoretically has the same complexity with the MMAS, but it is faster in simulation thanks to not having the evaporation for the edges after ants visited. And the MLAS algorithm does not need neither the evaporation for the edges nor the running for removing the unnecessary edges, then this algorithm have the best complexity with only $O(nc * k * m * n)$.

4 Experimental Results

We used ten the benchmark data sets for 10-machine and 10-job problem [11]. [10] claimed that there is no approximate algorithm for solving JSS problem which always output the result less than 5/4 of the optimal result. It meant 5/4 the optimal solution is the target of the heuristic approaches for this class of such problems, and the efficient algorithms should give the result around that criterion.

The out-performance in runtime complexity with our improvements is verified with the experimental results with the same benchmark data sets described above. For the fair comparison, we chose the same parameters for all considered algorithms : the evaporation rate $\rho = 0.01$, the number of run loops is $n_{loop} = 50000$, the number of used ant agents is $n_{ant} = 10$, and two scaling parameters $\alpha = 1.0$ and $\beta = 0.4$ for the probability of each ant to choose the next node in the graph. And we set the ratio 1/2 for τ_{min}/τ_{max} for all compared algorithms as discussed above (with MLAS, τ_{min} is replaced by τ_{mid}). And to emphasize the impacts of the pheromone update rules in ACO, we do not use any *local*

Table 1. Comparison in term of best result, where Opt is optimal result known in advance, MMAS stands for the Max-Min Ant System algorithm, SMMAS is our smooth Max-Min Ant System algorithm and MLAS is our multi-level ant system algorithm. The number at first columns stand for the corresponding benchmark tests Orb1, ..., Orb10 [11]. The number in the brackets indicates the percentage difference of the result of each specific algorithm from the 5/4 optimal result, we have the better result with the negative numbers. The **in-bold** result is the best result among three compared algorithms.

No.	5/4Opt	MMAS		SMMAS		MLAS	
1	1323.8	1340(1.2)	1368(3.3)	1329(0.4)	1363.3(3.0)	1328(0.3)	1357.8(2.6)
2	1110	1093(-1.5)	1127.8(1.6)	1090(-1.8)	1117.4(0.7)	1094(-1.4)	1121.2(1.0)
3	1256.3	1309(4.2)	1348.5(7.3)	1301(3.6)	1348.2(7.3)	1322(5.2)	1336.7(6.4)
4	1256.3	1248(0.5)	1297.2(3.3)	1224(-0.7)	1291.2(2.8)	1248(-0.7)	1291.6(2.8)
5	1108.8	1078(-2.8)	1118.3(0.9)	1086(-2.1)	1105.4(-0.3)	1086(-2.1)	1104.5(-0.4)
6	1262.5	1281(1.5)	1344.4(6.5)	1304(3.3)	1332.2(5.5)	1281(1.5)	1329.3(5.3)
7	496.3	510(2.8)	514.7(3.7)	509(2.6)	513.4(3.4)	498(2.3)	509(2.6)
8	1123.8	1179(4.9)	1189.9(5.9)	1128(0.4)	1173.8(4.4)	1150(2.3)	1164.7(3.6)
9	1167.5	1168(0.0)	1198.3(2.6)	1144(-1.7)	1195.6(2.4)	1169(0.1)	1190.1(1.9)
10	1180	1194(1.2)	1228.1(4.1)	1177(-0.3)	1207.1(2.3)	1187(0.6)	1219.9(3.4)

search for all of three compared algorithms. We also calculate the differences between the 5/4 optimal result and the result of each algorithm to show the efficiency of these compared methods in the general context. Table 1 represents the experimental results of all compared algorithms for the data sets described above. With each method, we have two values : the best so-far result and the average result. Our improved algorithms is outperformed in almost cases (except the data set Orb5, and have the same result with the existed MMAS in Orb6). The smooth MMAS have the best result in 6 over 10 data sets. Because ACO algorithm class is meta-heuristic framework, then the best result do not always guarantee for evaluating the results of proposed methods. Average results give us the more sensitive comparison. And in this sense, the improved system is totally better than the MMAS. And the multi-level system is outperformed in 7 out of 10 cases while the SMMAS is only for 3 data sets. This result suggested us to used both methods depending on the specific purpose. Actually, the ratio $1/2$ for τ_{mid}/τ_{max} is not really good control for pheromone trail update in MLAS. The balance between the random search and reinforcement from the pheromone trail information would be better if we have a more flexible control, but it is not easy to do for all problems [8].

5 Conclusion

Recent studies and applications in ACO do not show any insight into the speed of convergence properties for class of ACO algorithms. Hence, the performance of applications are still limited. Our papers gave some properties that showed the behavior of pheromone trail intensity at a specific loop. They are really

useful for applications. We verified the mathematical results in a typical type of JSS problem where the experimental results showed the efficiency besides our better complexity. Application of ACO in the general combinatorial optimization problems is still attractive. And our proposed improvements have a suggestion for obtaining the better results in solving the combinatorial optimization problems although adjusting the parameters still depending on the way to choose the pheromone trail and the type of heuristic information for each specific problem.

References

1. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Politecnico di Milano, Italy, Tech. Rep. 91–106 (1991)
2. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. *The IEEE Transactions on Systems, Man and Cybernetics, Part B* 26(1), 29–41 (1996)
3. Dorigo, M., Gambardella, L.: Ant colony system: a cooperative learning approach to the traveling salesman problem. *The IEEE Transactions on Evolutionary Computation* 1(1), 53–66 (1997)
4. Dorigo, M., Di Caro, M.: Ant colony optimization: a new metaheuristic. In: *The 1999 Congress on Evolutionary Computation (CEC 1999)*, vol. 2, pp. 6–9 (1999)
5. Stutzle, T., Hoos, H.: MAX-MIN ant system. *The Future Generation Computer Systems* 16(9), 889–914 (2000)
6. Stutzle, T., Dorigo, M.: A short convergence proof for a class of ant colony optimization algorithms. *The IEEE Transactions on Evolutionary Computation* 6(4), 358–365 (2002)
7. Gutjahr, W.J.: Mathematical runtime analysis of ACO algorithms: survey on an emerging issue. *Swarm. Intell.* 1, 59–79 (2007)
8. Huy, D.Q., Dong, D.D., Huan, H.X.: Multi-level ant system - a new approach through the new pheromone update for ant colony optimization. In: *The 2006 IEEE Conference on Research, Innovation and Vision for the Future*, pp. 55–58 (2006)
9. van der Zwaan, S., Marques, C.: Ant Colony Optimisation for Job Shop Scheduling. In: *Proceedings of the Third Workshop on Genetic Algorithms and Artificial Life (GAAL 1999)* (1999)
10. Vaessens, R., Aarts, E., Lenstra, J.: Job shop scheduling by local search. *INFORMS Journal on Computing*, vol 8, 302–317 (1996)
11. Applegate, D., Cook, W.: A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*, vol 3(1) (1991)

Preliminary Result on Secure Protocols for Multiple Issue Negotiation Problems

Katsuhide Fujita¹, Takayuki Ito^{1,2}, and Mark Klein²

¹ Techno-Business School Nagoya Institute of Technology Nagoya, Japan

² Center for Collective Intelligence Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA

Abstract. Multi-issue negotiation protocols represent a promising field since most negotiation problems in the real world involve multiple issues. Our work focuses on negotiation with multiple interdependent issues in which agent utility functions are nonlinear. Existing works have not yet concerned with agents' private information that should be concealed from others in negotiations. In this paper, we propose Distributed Mediator Protocol and Take it or Leave it Protocol for negotiation that can reach agreements and protect agents' private information. Moreover, we propose Hybrid Secure Protocol that combines Distributed Mediator Protocol with Take it or Leave it Protocol. The Hybrid Secure Protocol can also reach agreements while completely concealing agents' private information. Furthermore, the Hybrid Secure Protocol achieves high optimality and uses less memory.

1 Introduction

Multi-issue negotiation protocols represent an important field of study. Even though there has been much previous work in this area [1,2,3], most deal exclusively with simple negotiations involving independent multiple issues. Many real-world negotiation problems, however, are complex and involve interdependent multiple issues. Thus, we focus on complex negotiations with interdependent multiple issues. These previous studies mainly assume that agents have an incentive to cooperate to achieve win-win agreements because the situation is not a zero-sum game.

Existing works have not yet been concerned with agents' private information. In negotiation, agents' private information should not be revealed to other agents and mediators. For example, suppose that several companies collaboratively design and develop a new car model. If one company reveals more private information than the other companies, the other companies will know more of that company's important information, such as utility information. As a result, the company suffers a disadvantage in subsequent negotiations, and the mediator might leak the agent's utility information. Furthermore, explicitly revealing private information is dangerous for security reasons. Therefore, our aim is to create a protocol that will find high-quality solutions while concealing agent's utility information.

We previously proposed a bidding-based negotiation protocol that focuses on interdependent multiple issues. Agents generate bids by sampling and searching their utility functions, and the mediator finds the optimum combination of

submitted bids from agents [4]. This protocol can achieve an agreement without revealing all agent private information. Moreover, we proposed a threshold adjusting mechanism where the mediator adjusts the agent’s threshold for generating bids. In the threshold adjusting mechanism, agents make agreements without excessively revealing their utility information [5]. However, since in these protocols the computational complexity for finding the solution is too large, we proposed a representative-based protocol [6] where the mediator selects representatives who propose alternatives to other agents. This protocol drastically reduced the computational complexity of the number of agents because the number of agents that make agreements was reduced.

Though, there are two main issues in the above protocols. First, it is impossible for the above protocols to conceal all agent private information because agents have to reveal some private information. Additionally, scalability for the complexity of agent’s utility function isn’t very high. Therefore, we need to create another new protocol that conceals all agent private information with high scalability for the complexity of agent utility functions.

In this paper, we propose the Distributed Mediator Protocol (DMP) and the Take it or Leave it (TOL) Protocol. They make agreements and conceal agent utility values. In the Distributed Mediator Protocol, we assume many mediators who search in utility space to find agreements. When searching in their search space, they employ the Multi-Party Protocol with which they can simultaneously calculate the sum the per agent utility value and conceal it. Furthermore, DMP improves the scalability for the complexity of the utility space by dividing the search space toward the mediators. In the Take it or Leave it (TOL) Protocol, the mediator searches using the hill-climbing search algorithm. The evaluation value is decided by responses that agents either take or leave moving from the current state to the neighbor state.

Moreover, we propose the Hybrid Secure Protocol (HSP) that combines DMP with TOL. In HSP, TOL is performed first to improve the initial state in the DMP step. Next, DMP is performed to find the local optima in the neighborhood. HSP can also reach an agreement and conceal per agent utility information. Additionally, HSP can reduce the required memory for making an agreement, which is a major issue in DMP. Our experimental results show that HSP can improve memory usage more than DMP.

The remainder of the paper is organized as follows. First, we describe a model of nonlinear multi-issue negotiation. Second, we propose the Distributed Mediator Protocol (DMP) and the Take it or Leave it (TOL) Protocol. Third, we propose the Hybrid Secure Protocol (HSP). Fourth, we present the experimental results about optimality and memory. Finally, we describe related work and draw conclusions.

2 Negotiations with Nonlinear Utility Functions

We consider the situation where n agents want to reach an agreement with a mediator who manages the negotiation from the middle position. There are m

issues, $s_j \in S$, to be negotiated. The number of issues represents the number of the dimensions of the utility space. For example, if there are three issues¹, the utility space has three dimensions. Issue s_j has a value drawn from the domain of integers $[0, X]$, *i.e.*, $s_j \in [0, X](1 \leq j \leq M)$ ².

A contract is represented by a vector of issue values $\mathbf{s} = (s_1, \dots, s_m)$.

An agent's utility function is described in terms of constraints. There are l constraints, $c_k \in C$. Each constraint represents a region with one or more dimensions and has an associated utility value. Constraint c_k has value $w_i(c_k, \mathbf{s})$ if and only if it is satisfied by contract $\mathbf{s}(1 \leq k \leq l)$. Every agent has its own, typically unique, set of constraints.

An agent's utility for contract \mathbf{s} is defined as $u_i(\mathbf{s}) = \sum_{c_k \in C, \mathbf{s} \in x(c_k)} w_i(c_k, \mathbf{s})$, where $x(c_k)$ is a set of possible contracts (solutions) of c_k . This expression produces a "bumpy" nonlinear utility space with high points where many constraints are satisfied and lower regions where few or no constraints are satisfied. This represents a crucial departure from previous efforts on multi-issue negotiation, where contract utility is calculated as the weighted sum of the utilities for individual issues, producing utility functions shaped like flat hyperplanes with a single optimum.

Figure 1 shows an example of a binary constraint between Issues 1 and 2. Utility function A, which has a value of 55, holds if the value for Issue 1 is in the range $[3, 7]$ and the value for Issue 2 is in the range $[4, 6]$. The utility space is highly nonlinear with many hills and valleys. In real situations, we assume that their utility spaces are more complex because users have more utility functions.

We assume, as is common in negotiation contexts, that agents do not share their utility functions with each other to preserve a competitive edge. Generally, in fact, agents do not completely know their desirable contracts in advance, because their own utility functions are simply too large. If we have 10 issues with 10 possible values per issue, for example, this produces a space of 10^{10} (10 billion) possible contracts, which is too many to evaluate exhaustively. Agents must thus operate in a highly uncertain environment.

Finding an optimal contract for individual agents with such utility spaces can be handled using well-known nonlinear optimization techniques such as simulated annealing or evolutionary algorithms. We cannot employ such methods for negotiation purposes, however, because they require that agents fully reveal their utility functions to a third party, which is generally unrealistic in negotiation contexts.

¹ The issues are not "distributed" over agents, who are all negotiating a contract that has N (e.g., 10) issues in it. All agents are potentially interested in the values for all N issues.

² A discrete domain can come arbitrarily close to a real domain by increasing its size. As a practical matter, many real-world issues that are theoretically real (delivery date, cost) are discretized during negotiations. Our approach, furthermore, is not theoretically limited to discrete domains. The deal determination part is unaffected, although the bid generation step will have to be modified to use a nonlinear optimization algorithm suited to real domains.

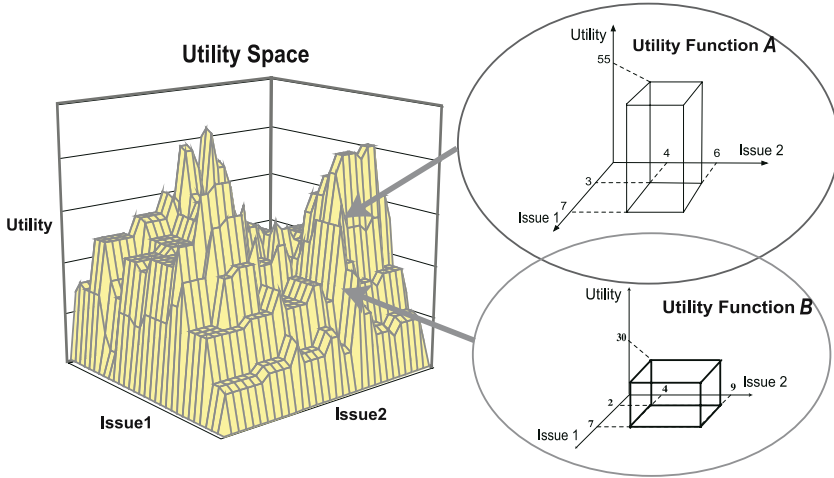


Fig. 1. Nonlinear utility function and complex utility space

The objective function for our protocol can be described as follows:

$$\arg \max_{\mathbf{s}} \sum_{i \in N} u_i(\mathbf{s}).$$

Our protocol, in other words, tries to find contracts that maximize social welfare, *i.e.*, the total utilities for all agents. Such contracts, by definition, will also be Pareto-optimal.

3 Secure and Scalable Negotiation Protocols

3.1 Distributed Mediator Protocol for Negotiation

We propose the Distributed Mediator Protocol (DMP) in this subsection. In DMP, there are more than two mediators (**Distributed Mediator**). DMP achieves security for agent’s private information by employing the Multi-Party Protocol [7]. Moreover, DMP achieves scalability for the utility space.

DMP is shown as follows:

We assume n mediators ($M_0, \dots, M_j, \dots, M_n$) who can calculate the sum of all the agent utility values if k mediators get together. Additionally, there are m agents ($Ag_0, \dots, Ag_i, \dots, Ag_m$). All mediators share q , which is preliminarily the prime number.

Step 1: The mediators divide the utility space (search space) and choose a mediator who manages it. How to divide the search space and assign tasks is beyond the scope of this present discussion. Parallel computation is possible by dividing the search space. This means that the computational complexity during searching can decrease.

Step 2: Each mediator searches her search space with a local search algorithm [8]. Hill-climbing search (HC) and simulated annealing search are examples of local search algorithms. During the search, the mediator declares a Multi-Party Protocol if he/she is searching in the state for the first time. After that, the mediator selects k mediators from all mediators and asks for generating v (shares) from all agents.

Step 3: Agent $i(A_i)$ randomly selects k dimension formula, which fulfills $f_i(0) = x_i$, and calculates $v_{i,j} = f_i(j)$. After that, agent i clandestinely sends $v_{i,j}$ to M_j .

Step 4: Mediator $j(M_j)$ receives $v_{1,j}, \dots, v_{m,j}$ from all agents. M_j calculates $v_j = v_{1,j} + \dots + v_{n,j} \text{ mod } q$ and reveals v_j to the other mediators.

Step 5: The mediators calculate $f(j)$, which fulfills $f(j) = v_j$ by Lagrange's interpolating polynomial. Finally, s , which fulfills $f(0) = s$, is the sum of all agent utility values.

Steps 2 ~ 5 are repeated until they fulfill the at-end condition in the local search algorithm. Finally, each mediator informs the maximum value (alternative) in his space to all mediators. After that, the mediators select the maximum value from all alternatives.

Figure 2 shows the flow in DMP. There are three agents and two mediators. If two mediators get together, they can calculate the sum of the per agent utility value ($k = n$). The gray area shows that agents perform the steps without revealing them. As the figure indicates, the selection of multinomial (f_i), generating share (v), adding the share, and Lagrange's interpolating polynomial can calculate the sum of all agent utility values and conceal them.

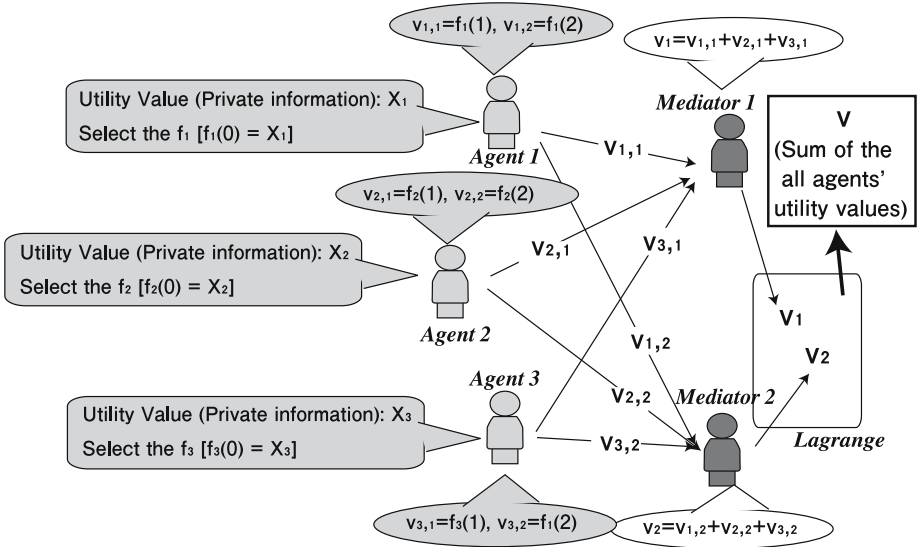


Fig. 2. Distributed Mediator Protocol

DMP has a security advantage and scalability for utility space. The details are shown as follows.

[Security]

DMP can calculate the sum of all agent utility values and conceal them. The proof is identical as the Multi-Party Protocol [7]. In DMP, other agents and the mediators can't know the agent utility values without illegally colluding.

Additionally, k , which is the number of mediators performing the multi-party protocol, is the tradeoff between security and computational complexity. If k mediators exchange their shares (v) illegally, they can expose the agent utility values. Therefore, it is good for security that k is such a large number that mediators can't collude illegally. However, it requires more computation time because more mediators have to stop searching.

[Scalability]

The computational cost can be greatly reduced because the mediators divide the search space. In existing protocols, they can't find better agreements when the search space becomes too large. However, this protocol can locate better agreements in large search spaces by dividing the search space.

DMP has a weak point: too many shares (v) are generated. This is because shares are generated that correspond to the search space. To generate shares takes more time than searching without generating shares. Thus, we need to generate fewer shares with high optimality.

3.2 Take It or Leave It (TOL) Protocol for Negotiation

In this subsection, we propose the *Take it or Leave it (TOL) Protocol*, which can also reach agreements and conceal all agents' utility information. The mediator searches with the hill-climbing search algorithm [8], which is a simple loop that continuously moves in the direction of increasing evaluated value [8]. Additionally, evaluated value is determined by the responses that agents take or leave to the offers to move from the current state to the neighbor state. The agents can conceal their utility value using this evaluation value.

This protocol consists of following steps.

Step1: The mediator randomly selects the initial state.

Step 2: The mediator asks the agents to move from the current to the neighbor state.

Step 3: Each agent compares its current state with the neighbor state and determines whether to take or leave it. If the neighbor state provides higher utility value than the current state, the agent "takes it". If the current state provides higher or identical utility value than the neighbor state, the agent "leaves it".

Step4: The mediator selects the next state declared by the most agents as "take it". However, the mediator selects the next state randomly if there are more than two states that most agents declared as "take it". The mediator can prevent the local maxima from being reached by random selection. Steps 2, 3, and 4 are repeated until all agents declare "leave it" or the mediator determines that a

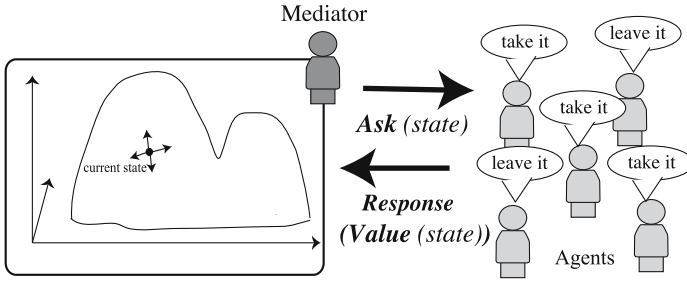


Fig. 3. Take it or Leave it (TOL) Protocol

plateau has been reached. A plateau is an area of the state space landscape where the evaluation function is flat.

Figure 3 shows the concept of the “Take it or Leave it (TOL) Protocol”. First, the mediator informs agents about the state whose evaluation value he wants to know. Second, agents search for their utility space and declare “take it” or “leave it”. Then they tell the number of agents who declare “take it” ($VALUE(state)$). These steps are repeated until they satisfy the at-end condition.

The “Take it or Leave it (TOL) Protocol” has an advantage of lower time complexity because it easily rates evaluated value. However, this protocol can’t find high optimality solutions when a plateau is reached.

4 Hybrid Secure Protocol(HSP) for Negotiation

Since there are weak points in DMP and TOL, as proposed in the previous section, we propose a new protocol that combines DMP with TOL. This new protocol is called the Hybrid Secure Protocol (HSP) for negotiation.

The Hybrid Secure Protocol (HSP) is shown as follows:

Step 1: The mediators divide the utility space (search space) and choose a mediator who manages it.

Step 2: Each mediator searches in her search space using TOL proposed in 3.2. The initial state is selected randomly. By performing the TOL at first, the mediators can find somewhat higher optimality of solutions without generating shares (v).

Step 3: Each mediator searches in her search space using DMP proposed in 3.1. The initial state is the solution found in Step 2. By performing DMP after TOL, mediators can find the local optima in the neighborhood and conceal the per agent private information.

Steps 2 and 3 are repeated many times by changing the initial state.

HSP can find solutions with fewer shares than DMP because the initial state in Step 3 is higher than only performing DMP. In addition, TOL doesn’t generate shares, and DMP searches in states in which TOL hasn’t searched. Thus, HSP can reduce the number of shares.

Meanwhile, optimality in HSP is higher. TOL usually stops searching after reaching the plateau. Additionally, the main reason for lowering the optimality in DMP is to reach the local optima, although the initial value in Step 3 is usually different because it is decided by TOL. Therefore, HSP can find higher agreement in optimality.

5 Experiment Results

5.1 Setting of Experiment

We conducted several experiments to evaluate the effectiveness of our approach. In each experiment, we ran 100 negotiations between agents with randomly generated utility functions.

In the optimality experiments, for each run, we applied an optimizer to the sum of all agent utility functions to find the contract with the highest possible social welfare. This value was used to assess the efficiency (*i.e.*, how closely optimal social welfare was approached) of the negotiation protocols. To find the optimum contract, we used simulated annealing (SA) because exhaustive search became intractable as the number of issues grew too large. The SA initial temperature was 50.0, which decreased linearly to 0 over the course of 2500 iterations. The initial contract for each SA run was randomly selected.

The following are the parameters for our experiments:

The number of agents is six, and the number of mediators is $2^{(\text{Number of Issues})}$. In DMP, they can calculate the sum of the per agent utility values if four mediators get together. In DMP, the search space is divided equally.

[Setting the utility function]

The domain for the issue values is $[0, 9]$. Constraints include 10 unary constraints, 5 binary constraints, 5 trinary constraints, etc. (a unary constraint relates to one issue, a binary constraint relates to two issues, and so on). The maximum value for a constraint is $100 \times (\text{Number of Issues})$. Constraints that satisfy many issues have, on average, larger weights, which seems reasonable for many domains. To meet scheduling, for example, higher order constraints concern more people than lower order constraints, so they are more important. The maximum width for a constraint is 7. The following constraints, therefore, would all be valid: Issue 1 = $[2, 6]$, Issue 3 = $[2, 9]$, and Issue 7 = $[1, 3]$.

[Setting of Simulated Annealing (SA)]

The annealing schedule for the distributed mediator protocol included an initial temperature of 20 with 5000 iterations. Note that the annealer must not run too long or too ‘hot’ because then each initial state by TOL will tend to find the global optimum instead of the peak of the optimum nearest the initial state in DMP.

In our experiments, we ran 100 negotiations in every condition. Our code was implemented in Java 2 (1.5) and run on a core 2 duo processor iMac with 1.0 GB memory on a Mac OS X 10.5 operating system.

5.2 Experimental Results

Figures 4 and 5 compare the protocols proposed in this paper. “(A) DMP (SA)” is the distributed mediator protocol, and the search algorithm is simulated annealing [8]. “(B) DMP (HC)” is the distributed mediator protocol, and the search algorithm is the hill-climbing algorithm [8]. “(C) HSP (SA)” is the hybrid secure protocol, and the search algorithm in the distributed mediator step is simulated annealing. “(D) HSP (HC)” is the hybrid secure protocol, and the search algorithm in the distributed mediator step is the hill-climbing algorithm. “(E) TOL” is the Take it or Leave it Protocol only.

Figure 4 shows the optimality rate in five protocols. “(B) DMP (HC)” decreases rapidly based on the number of issues because hill-climbing reaches local optima by increasing the search space. Additionally, “(A) DMP (SA)” is the same as the optimal solution. Therefore, optimality in DMP depends on the search algorithm. “(E) TOL” is stable, so it does not decrease rapidly because each agent tries to find a better state in each utility space. “(C) HSP (SA)” and “(D) HSP (HC)” have higher optimality than “(E) TOL” because HSP performs DMP after performing TOL. In addition, “(D) HSP (HC)” has higher optimality than “(C) HSP (SA)” because SA in the DMP step sometimes stops searching for a worse state than the initial state due to a random nature. But HC stops searching for a better state than the initial state.

Figure 5 shows the average share (v) per agent. The number of shares shows a comparison of memory in several protocols. “(A) DMP (SA)” increases exponentially. On the other hand, “(B) DMP (HC)” reduces the shares compared to “(A) DMP (SA)” because SA searches for more states than HC. The number of shares in DMP depends on the features of the search protocol. Furthermore, “(C)

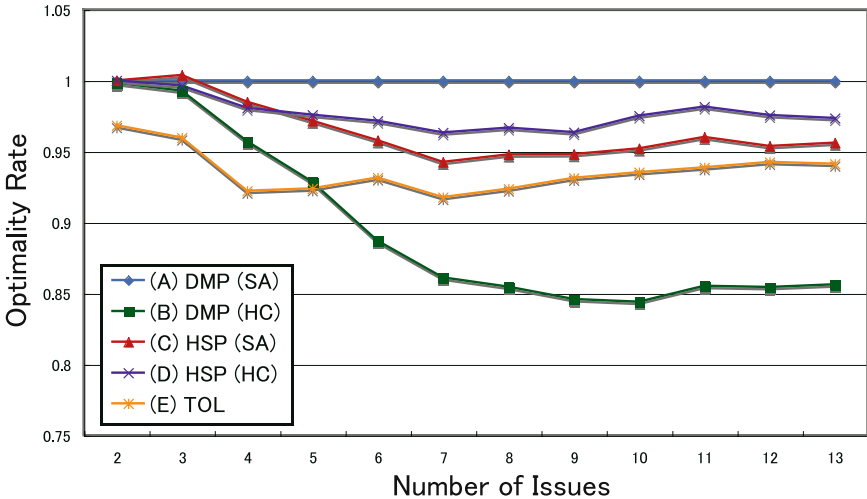


Fig. 4. Optimality Rate

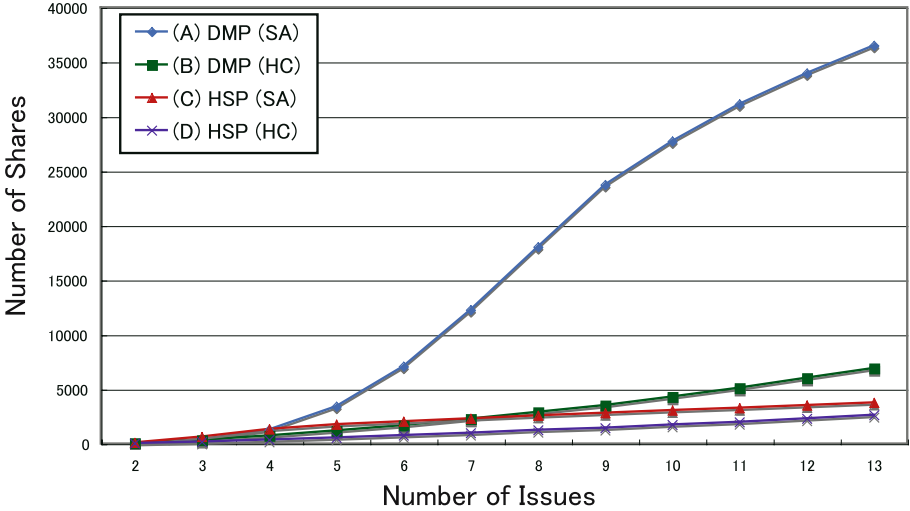


Fig. 5. The number of shares

HSP (SA)” and “(D) HSP (HC)” reduce shares compared to “(A) DMP (SA)” and “(B) DMP (HC)” because the initial state in the DMP step in HSP has a higher value than the initial state in DMP since TOL was performed before. Thus, HSP can reduce the shares more than DMP.

From the above experiments HSP can reduce the shares with high optimality.

6 Related Work

Most previous work on multi-issue negotiation [12,3] has only addressed linear utilities. Recently some researchers have been focusing on more complex and nonlinear utilities. [9] has explored a range of protocols based on mutation and selection on binary contracts. This paper does not describe what kind of utility function is used, nor does he present any experimental analyses, it is unclear whether this strategy enables sufficient exploration of utility space. [10] presents an approach based on constraint relaxation. However, there is no experimental analysis, and this paper merely presents a small toy problem with 27 contracts. [11] modeled a negotiation problem as a distributed constraint optimization problem. This paper claims the proposed algorithm is optimal, but it does not discuss computational complexity and only provides a single small-scale example.

Based on a simulated-annealing mediator, [12] presented a protocol that was applied with near-optimal results to medium-sized bilateral negotiations with binary dependencies. The work presented here is distinguished by demonstrating both scalability and high optimality values for multilateral negotiations and higher order dependencies. [13,14] also presented a protocol for multi-issue problems for bilateral negotiations. [15,16] presented a multi-item and multi-issue

negotiation protocol for bilateral negotiations in electronic commerce situations. [17] proposed bilateral multi-issue negotiations with time constraints, and [18] proposed multi-issue negotiation that employs a third-party to act as a mediator to guide agents toward equitable solutions. This framework also employs an agenda that serves as a schedule for the ordering of issue negotiation. Agendas are very interesting because agents only need to focus on a few issues. [19] proposed a checking procedure to mitigate this risk and show that by tuning this procedure's parameters, outcome deviation can be controlled. These studies reflect interesting viewpoints, but they focused on just bilateral trading or negotiations.

7 Conclusion

In this paper, we proposed the Distributed Mediator Protocol (DMP) and the Take it or Leave it (TOL) Protocol that can reach agreements and conceal agent's utility information and achieve high scalability in utility space. Moreover, we proposed the Hybrid Secure Protocol (HSP) that combines DMP and TOL. Experimental results demonstrated that HSP can reduce memory with high optimality.

One future work includes a method to divide the search space depending on agent power. A protocol that develops the scalability of utility information is also possible future work. One possible protocol is to break up the agenda of issues.

References

1. Bosse, T., Jonker, C.M.: Human vs. computer behaviour in multi-issue negotiation. In: Proc. of 1st International Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems (RRS 2005), pp. 11–24 (2005)
2. Faratin, P., Sierra, C., Jennings, N.R.: Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence* 142, 205–237 (2002)
3. Fatima, S., Wooldridge, M., Jennings, N.R.: Optimal negotiation of multiple issues in incomplete information settings. In: Proc. of Third International Joint Conference on Autonomous Agent and Multi-Agent Systems (AAMAS 2004), pp. 1080–1087 (2004)
4. Ito, T., Hattori, H., Klein, M.: Multi-issue negotiation protocol for agents: Exploring nonlinear utility spaces. In: Proc. of 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 1347–1352 (2007)
5. Fujita, K., Ito, T., Hattori, H., Klein, M.: An approach to implementing a threshold adjusting mechanism in very complex negotiations: A preliminary result. In: Proc. of The 2nd International Conference on Knowledge, Information and Creativity Support Systems (KICSS 2007) (2007)
6. Fujita, K., Ito, T., Klein, M.: A representative-based multi-round protocol for multi-issue negotiations. In: Proc. of th 7th International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2008) (2008)
7. Lindell, Y.: *Composition of Secure Multi-Party Protocols: A Comprehensive Study*. Springer, Heidelberg (2003)

8. Russell, S.J., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs (2002)
9. Lin, R.J., Chou, S.T.: Bilateral multi-issue negotiations in a dynamic environment. In: *Proc. of AMEC 2003* (2003)
10. Barbuceanu, M., Lo, W.K.: Multi-attribute utility theoretic negotiation for electronic commerce. In: Dignum, F.P.M., Cortés, U. (eds.) *AMEC 2000*. LNCS, vol. 2003, pp. 15–30. Springer, Heidelberg (2001)
11. Luo, X., Jennings, N.R., Shadbolt, N., Leung, H., Lee, J.H.: A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. *Artificial Intelligence* 148, 53–102 (2003)
12. Klein, M., Faratin, P., Sayama, H., Bar-Yam, Y.: Negotiating complex contracts. *Group Decision and Negotiation* 12(2), 58–73 (2003)
13. Lai, G., Li, C., Sycara, K.: A general model for pareto optimal multi-attribute negotiations. In: *Proc. of The 2nd International Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems (RRS 2006)* (2006)
14. Lai, G., Sycara, K., Li, C.: A decentralized model for multi-attribute negotiations with incomplete information and general utility functions. In: *Proc. of The 2nd International Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems (RRS 2006)* (2006)
15. Robu, V., Poutre, H.L.: Retrieving the structure of utility graphs used in multi-item negotiation through collaborative filtering of aggregate buyer preferences. In: *Proc. of The 2nd International Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems (RRS 2006)* (2006)
16. Robu, V., Somefun, D.J.A., Poutre, J.L.: Modeling complex multi-issue negotiations using utility graphs. In: *Proc. of the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005)* (2005)
17. Fatima, S.S., Wooldridge, M., Jennings, N.R.: Approximate and online multi-issue negotiation. In: *Proc. of the 6th International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2007)*, pp. 947–954 (2007)
18. Shew, J., Larson, K.: The blind leading the blind: A third-party model for bilateral multi-issue negotiations under incomplete information. In: *Proc. of The 1st International Workshop on Agent-based Complex Automated Negotiations (ACAN 2008)* (2008)
19. Hindriks, K., Jonker, C., Tykhonov, D.: Avoiding approximation errors in multi-issue negotiation with issue dependencies. In: *Proc. of The 1st International Workshop on Agent-based Complex Automated Negotiations (ACAN 2008)* (2008)

Performance Analysis about Parallel Greedy Approximation on Combinatorial Auctions

Naoki Fukuta¹ and Takayuki Ito^{2,3}

¹ Shizuoka University, Hamamatsu Shizuoka 4328011, Japan
fukuta@cs.inf.shizuoka.ac.jp

<http://whitebear.cs.inf.shizuoka.ac.jp/>

² Nagoya Institute of Technology, Gokiso-cho Nagoya 4668555, Japan

³ Massachusetts Institute of Technology, Cambridge, MA 02142 USA

Abstract. Combinatorial auctions provide suitable mechanisms for efficient allocation of resources to self-interested agents. Considering ubiquitous computing scenarios, the ability to complete an auction within a fine-grained time period without loss of allocation efficiency is in strong demand. Furthermore, to achieve such scenarios, it is very important to handle a large number of bids in an auction. Recently, we proposed an algorithm to obtain sufficient quality of winners in very short time. However, it is demanded to analyze which factor is mainly affected to obtain such a good performance. Also it is demanded to clarify the actual implementation-level performance of the algorithm compared to a major commercial-level generic problem solver. In this paper, we show our parallel greedy updating approach contributes its better performance. Furthermore, we show our approach has a certain advantage compared to a latest commercial-level implementation of generic LP solver through various experiments.

1 Introduction

Combinatorial auctions [1], one of the most popular market mechanisms, have a huge effect on electronic markets and political strategies. For example, Sandholm et al. [2] proposed a market using their innovative combinatorial auction algorithms. Combinatorial auctions provide suitable mechanisms for efficient allocation of resources to self-interested attendees [1]. Therefore, many works have been done to utilize combinatorial auction mechanisms for efficient resource allocation. For example, the FCC tried to employ combinatorial auction mechanisms for assigning spectrums to companies [3].

On the other hand, efficient resource allocation is also becoming crucial in many computer systems that should manage resources efficiently, and combinatorial auction mechanisms are suitable for this situation. For example, considering a ubiquitous computing scenario, there is typically a limited amount of resources (sensors, devices, etc.) that may not cover all needs for all agents. Due to certain reasons (physical limitations, privacy, etc.), most of the resources cannot be shared with other agents. Furthermore, agents will use two or more

resources at a time to achieve desirable services for users. Of course, each agent provides services to its own user, and the agent may be self-interested.

We proposed our initial idea of winner approximation algorithm [4], presented comparative evaluation of algorithms [5] [6], proposed an extension for effective reuse among similar problems [7], and discussed theoretical issues in such approximation [8]. However, it is demanded to analyze which factor is mainly affected to obtain such a good performance. Also it was unclear how good the actual implementation-level performance of the algorithm is, comparing to a major commercial-level generic problem solver. In this paper, we analyze the performance of our algorithm with slightly modified variants on numerous datasets. Furthermore, we show our approach has a certain advantage compared to a latest commercial-level implementation of generic linear problem(LP) solver.

2 Preliminaries

2.1 Winner Determination Problem

The winner determination problem on combinatorial auctions is defined as follows [1]: The set of bidders is denoted by $N = 1, \dots, n$, and the set of items by $M = \{m_1, \dots, m_k\}$. $|M| = k$. Bundle S is a set of items: $S \subseteq M$. We denote by $v_i(S)$, bidder i 's valuation of the combinatorial bid for bundle S . An allocation of the items is described by variables $x_i(S) \in \{0, 1\}$, where $x_i(S) = 1$ if and only if bidder i wins bundle S . An allocation, $x_i(S)$, is feasible if it allocates no item more than once,

$$\sum_{i \in N} \sum_{S \ni j} x_i(S) \leq 1$$

for all $j \in M$. The winner determination problem is the problem to maximize total revenue

$$\max_X \sum_{i \in N, S \subseteq M} v_i(S) x_i(S)$$

for feasible allocations $X \ni x_i(S)$.

2.2 Lehmann's Greedy Winner Determination

Lehmann's greedy algorithm [9] is a very simple but powerful linear algorithm for winner determination in combinatorial auctions. Here, a bidder declaring $\langle s, a \rangle$, with $s \subseteq M$ and $a \in \mathcal{R}_+$ will be said to put out a bid $b = \langle s, a \rangle$. Two bids $b = \langle s, a \rangle$ and $b' = \langle s', a' \rangle$ conflict if $s \cap s' \neq \emptyset$. The greedy algorithm can be described as follows. (1) The bids are sorted by some criterion. The researchers [9] proposed sorting list L by descending average amount per item. More generally, they proposed sorting L by a criterion of the form $a/|s|^c$ for some number c , $c \geq 0$, possibly depending on the number of items, k . (2) A greedy algorithm generates an allocation. L is the sorted list in the first phase. Walk down the list L , allocates items to bids whose items are still unallocated.

3 Enhanced Approximation Algorithms

3.1 Incremental Updating

In [4], we have shown that the hill-climbing approach performs well when an auction has a massively large number of bids. In this section, we summarize our proposed algorithms for incrementally updating solutions.

Lehmann's greedy winner determination could succeed in specifying the lower bound of the optimality in its allocation [9]. A straightforward extension of the greedy algorithm is to construct a local search algorithm that continuously updates the allocation so that the optimality is increased. Intuitively, one allocation corresponds to one state of a local search.

The inputs are *Alloc* and *L*. *L* is the bid list of an auction. *Alloc* is the initial greedy allocation of items for the bid list.

```

1: function GreedyHillClimbingSearch(Alloc, L)
2:   RemainBids := L - Alloc;
3:   for each b ∈ RemainBids as sorted order
4:     if b conflicts Alloc then
5:       Conflicted := Alloc - consistentBids({b}, Alloc);
6:       NewAlloc := Alloc - Conflicted + {b};
7:       ConsBids :=
8:         consistentBids(NewAlloc, RemainBids);
9:       NewAlloc := NewAlloc + ConsBids;
10:    if price(Alloc) < price(NewAlloc) then
11:      return GreedyHillClimbingSearch(NewAlloc, L);
12:  end for each
13:  return Alloc

```

The function *consistentBids* finds consistent bids for the set *NewAlloc* by walking down the list *RemainBids*. Here, a new inserted bid will wipe out some bids that conflict with the inserted bid. So there will be free items to allocate after the insertion. The function *consistentBids* tries to insert the other bids greedily for selling as many of the items as possible. When the total price for *NewAlloc* is higher than *Alloc*, current allocation is updated to *NewAlloc* and the function continues updating from *NewAlloc*. We call this as *Greedy Hill Climbing*(GHC) in this paper.

Also we prepared an ordinary Hill-Climbing local search algorithm. The difference to above is to choose 'best' alternatives in each climbing step, instead of choosing it greedily. We call this as *Best Hill Climbing*(BHC) in this paper.

```

1: function BestHillClimbingSearch(Alloc, L)
2:   MaxAlloc :=  $\phi$ 
3:   RemainBids := L - Alloc;
4:   for each b ∈ RemainBids as sorted order

```

```

5:   if  $b$  conflicts  $Alloc$  then
6:      $Conflicted := Alloc - consistentBids(\{b\}, Alloc)$ ;
7:      $NewAlloc := Alloc - Conflicted + \{b\}$ ;
8:      $ConsBids :=$ 
9:       consistentBids( $NewAlloc$ ,  $RemainBids$ );
10:     $NewAlloc := NewAlloc + ConsBids$ ;
11:   if  $price(MaxAlloc) < price(NewAlloc)$  then
12:      $MaxAlloc := NewAlloc$ ;
13:   end for each
14:   if  $price(Alloc) < price(MaxAlloc)$  then
15:     return BestHillClimbingSearch( $MaxAlloc, L$ );
16:   return  $Alloc$ 

```

3.2 Parallel Search for Multiple Weighting Strategies

The optimality of allocations got by Lehmann's algorithm (and the following hill-climbing) deeply depends on which value was set to c in the bid weighting function. Again, in [9], Lehmann et al. argued that $c = 1/2$ is the best parameter for approximation when the norm of the worst case performance is considered. However, optimal value for approximating an auction is varied from 0 to 1 depending on the auction problem. In [4], we presented an initial idea of an enhancement for our incremental updating algorithm to parallel search for different bid weighting strategies (e.g, doing the same algorithm for both $c = 0$ and $c = 1$).

4 Evaluation

4.1 Experiment Settings

In this section, we compare our algorithms to other approaches in various datasets. Details about other approaches are presented in section 5.

We implemented our algorithms in a C program for the following experiments. We also implemented the Casanova algorithm [10] in a C program. However, for the following experiments, for Zurel's algorithm we used Zurel's C++ based implementation that is shown in [11]. Also we used CPLEX Interactive Optimizer 11.0.0 (32bit) in our experiments.

The experiments were done with the above implementations to examine the performance differences among algorithms. The programs were employed on a Mac with Mac OS X 10.4, CoreDuo 2.0GHz CPU, and 2GBytes of memory. Thus, actual computation time will be much smaller when we employ parallel processor systems in a distributed execution environment. We leave this for future work.

We conducted several experiments. In each experiment, we compared the following search algorithms. `greedy(C=0.5)` uses Lehmann's greedy allocation algorithm with parameter ($c = 0.5$). `greedy-N` uses the best results of Lehmann's greedy allocation algorithm for N different weighting parameters ($0 \leq c \leq 1$). `*HC(c=0.5)` uses a local search in which the initial allocation is Lehmann's

Table 1. Optimality on CATS-VARSIZE dataset

	arbitrary	L2	L3	L4	L6	L7	matching	regions	scheduling	average
greedy($C=0.5$)	0.8641	0.9968	0.8037	0.9076	0.9403	0.8652	0.9721	0.8734	0.9143	0.9042
GHC($C=0.5$)	0.9485	1.0000	0.9433	0.9611	0.9902	0.9822	0.9957	0.9703	0.9826	0.9749
BHC($C=0.5$)	0.9441	1.0000	0.9435	0.9575	0.9895	0.9841	0.9960	0.9686	0.9860	0.9744
greedy-3	0.8809	0.9999	0.8237	0.9201	0.9571	0.8917	0.9742	0.8859	0.9474	0.9201
GHC-3	0.9669	1.0000	0.9568	0.9773	0.9947	0.9890	0.9964	0.9808	0.9930	0.9839
BHC-3	0.9640	1.0000	0.9562	0.9756	0.9948	0.9903	0.9967	0.9793	0.9948	0.9835
greedy-11	0.8973	1.0000	0.8359	0.9355	0.9683	0.9010	0.9749	0.9028	0.9664	0.9314
GHC-11	0.9750	1.0000	0.9663	0.9807	0.9957	0.9920	0.9967	0.9857	0.9975	0.9877
BHC-11	0.9712	1.0000	0.9660	0.9791	0.9958	0.9925	0.9970	0.9844	0.9969	0.9870
SA	0.9768	1.0000	0.9756	0.9813	0.9950	0.9921	0.9975	0.9872	0.9969	0.9892
Zurel	0.9671	0.9998	0.9571	0.9811	0.9977	0.9838	0.9994	0.9836	0.9909	0.9845
Casanova	0.9567	1.0000	0.9741	0.96457	0.9753	0.9905	0.9946	0.9711	0.9979	0.9805

allocation with $c = 0.5$ and conducts one of hill-climbing searches (e.g., GHC or BHC) shown in the previous section. Similarly, *HC-N uses the best results of a hill-climbing search (e.g.,GHC or BHC) for N different weighting parameters ($0 \leq c \leq 1$). For example, GHC-11 means the best result of greedy hill-climbing(GHC) with parameter $c = \{0, 0.1, \dots, 0.9, 1\}$. SA uses the simulated annealing algorithm presented in [4]. Also, we denote the Casanova algorithm as *casanova* and Zurel’s algorithm as *Zurel*.

In the following experiments, we used 0.2 for the epsilon value of Zurel’s algorithm in our experiments. This value appears in [11]. Also, we used 0.5 for np and 0.15 for wp on Casanova, which appear in [10]. Note that we set $maxTrial$ to 1 but $maxSteps$ to ten times the number of bids in the auction.

4.2 Evaluation on Basic Auction Dataset

In [11], the researchers evaluated the performance of their presented algorithm with the data set presented in [12], compared with CPLEX and other existing implementations. In [6], we presented comparison of our algorithms, Casanova, and Zurel’s algorithm with the dataset provided in [12]. This dataset contains 2240 auctions with optimal values, ranging from 25 to 40 items and from 50 to 2000 bids.

We conducted detailed comparisons with common datasets from CATS benchmark [13]. Compared to deVries’ dataset shown in [12], the CATS benchmark is very common and it contains more complex and larger datasets.

Table 1 shows the comparison of our algorithms, Casanova, and Zurel’s algorithm with a dataset provided in the CATS benchmark [13]. The dataset has numerous auctions with optimal values in several distributions. Here we used ‘varsize’ which contains a total of 7452 auctions with reliable optimal values in 9 different distributions.¹ Numbers of items range from 40 to 400 and numbers of bids range from 50 to 2000. The name of each distribution is referred from [13].

¹ Since some of the original data seems corrupted or failed to obtain optimal values, we excluded such auction problems from our dataset. Also, we excluded a whole dataset of a specific bid distribution when the number of valid optimal values is smaller than the other half of the data. The original dataset provides optimal values of auction problems by two independent methods, CASS and CPLEX. Therefore, it is easy to find out such corrupted data from the dataset.

Since problems in the dataset have relatively small size of bids and items, we omitted the execution time since all algorithms run in very short time. Here, we can see that the performances of GHC-11 and SA are better than Zurel's on arbitrary, L2, L3, L7, regions, and scheduling. Others are nearly equal to Zurel's. The performance of Casanova is nearly equal to or less than GHC(C=0.5) excluding L3 and scheduling.

Note that those differences come from the differences of the termination condition on each algorithm. In particular, Casanova spent much more time compared with the other two algorithms. However, we do not show the time performance here since the total execution time is relatively too small to be compared.

Here, We can see the performance of both greedy, GHC, and BHC increases when we use more threads to parallel search for multiple weightings. For example, the result of GHC-3 is better than GHC(c=0.5) and GHC-11 is slightly better in the average. It shows that our parallel approximation approach will increase the performance effectively even when the number of searching threads is small.

Also we compared the performance on our greedy local updating approach (GHC) with ordinary best updating approach(BHC). Surprisingly, the average performance of GHC are slightly better than BHC, regardless of using parallel search. This is because the BHC approach is still heuristic one so it does not guarantee the choice is best for global optimization. Also we think we found a very good heuristic bid weighting function for our greedy updating.

4.3 Evaluation on Large Auction Dataset

The CATS common datasets we used in Section 4.2 have a relatively smaller number of bids than we expected. We conducted additional experiments with much greater numbers of bids. We prepared additional datasets having 20,000 non-dominated bids in an auction. The datasets were produced by CATS [13] with default parameters in 5 different distributions. In the datasets, we prepared 100 trials for each distribution. Each trial is an auction problem with 256 items and 20,000 bids²

Table 2 shows the experimental result on the datasets with 20,000 bids in an auction focused on execution time of approximation. Due to the difficulty of attaining optimal values, we normalized all values as Zurel's results equaling 1 as follows.

Let A be a set of algorithms, $z \in A$ be the Zurel's approximation algorithm, L be a dataset generated for this experiment, and $revenue_a(p)$ such that $a \in A$ be the revenue obtained by algorithm a for a problem p such that $p \in L$, the average revenue ratio $ratioA_a(L)$ for algorithm $a \in A$ for dataset L is defined as follows:

$$ratioA_a(L) = \frac{\sum_{p \in L} revenue_a(p)}{\sum_{p \in L} revenue_z(p)}$$

Here, we use $ratioA_a(L)$ for our comparison of algorithms.

² Due to the difficulty of preparing the dataset, we only prepared 5 distributions. For more details about the bid generation problem, see [13]. A preliminary result of this experiment was shown in [5].

Table 2. Time Performance on 20,000 bids-256 items

	L2		L3		L4		L6		L7		average	
greedy($c=0.5$)	1.0002	(23.0)	0.9639	(19.0)	0.9417	(23.0)	0.9389	(23.4)	0.7403	(22.1)	0.9170	(22.1)
greedy-3-seq	1.0003	(69.1)	0.9639	(59.2)	0.9999	(72.9)	0.9965	(67.8)	0.7541	(66.8)	0.9429	(67.2)
greedy-3-para	1.0003	(26.4)	0.9639	(20.9)	0.9999	(28.4)	0.9965	(26.0)	0.7541	(25.5)	0.9429	(25.4)
BHC($c=0.5$)-100ms	1.0002	(100)	0.9639	(100)	0.9417	(100)	0.9389	(100)	0.7413	(100)	0.9170	(100)
BHC-3-seq-100ms	1.0003	(100)	0.9639	(100)	0.9999	(100)	0.9965	(100)	0.7541	(100)	0.9429	(100)
BHC-3-para-100ms	1.0003	(100)	0.9639	(100)	0.9999	(100)	0.9965	(100)	0.7541	(100)	0.9429	(100)
GHC($c=0.5$)-100ms	1.0004	(100)	0.9741	(100)	0.9576	(100)	0.9533	(100)	0.8260	(100)	0.9423	(100)
GHC-3-seq-100ms	1.0004	(100)	0.9692	(100)	1.0000	(100)	0.9966	(100)	0.8287	(100)	0.9590	(100)
GHC-3-para-100ms	1.0004	(100)	0.9743	(100)	1.0001	(100)	0.9969	(100)	0.9423	(100)	0.9828	(100)
BHC($c=0.5$)-1000ms	1.0002	(1000)	0.9639	(1000)	0.9417	(1000)	0.9389	(1000)	0.7413	(1000)	0.9170	(1000)
BHC-3-seq-1000ms	1.0003	(1000)	0.9639	(1000)	0.9999	(1000)	0.9965	(1000)	0.7541	(1000)	0.9429	(1000)
BHC-3-para-1000ms	1.0003	(1000)	0.9639	(1000)	0.9999	(1000)	0.9965	(1000)	0.7541	(1000)	0.9429	(1000)
GHC($c=0.5$)-1000ms	1.0004	(1000)	0.9856	(1000)	0.9771	(1000)	0.9646	(1000)	1.0157	(1000)	0.9887	(1000)
GHC-3-seq-1000ms	1.0004	(1000)	0.9804	(1000)	1.0003	(1000)	0.9976	(1000)	1.0086	(1000)	0.9975	(1000)
GHC-3-para-1000ms	1.0004	(1000)	0.9856	(1000)	1.0006	(1000)	0.9987	(1000)	1.0240	(1000)	1.0019	(1000)
SA-1200ms	1.0004	(1200)	0.9773	(1200)	0.9594	(1200)	0.9449	(1200)	1.0083	(1200)	0.9781	(1200)
Zurel-1st	0.5710	(11040)	0.9690	(537)	0.9983	(2075)	0.9928	(1715)	0.6015	(1796)	0.8265	(3433)
Zurel	1.0000	(13837)	1.0000	(890)	1.0000	(4581)	1.0000	(4324)	1.0000	(3720)	1.0000	(5470)
casanova-10ms	0.2583	(10)	0.0069	(10)	0.0105	(10)	0.0202	(10)	0.2577	(10)	0.0632	(10)
casanova-100ms	0.2583	(100)	0.0069	(100)	0.0105	(100)	0.0202	(100)	0.2577	(100)	0.1107	(100)
casanova-1000ms	0.5357	(1000)	0.1208	(1000)	0.0861	(1000)	0.1486	(1000)	0.7614	(1000)	0.3305	(1000)
cplex-100ms	0.0000	(288)	0.0000	(121)	0.0299	(111)	0.0000	(150)	0.0000	(119)	0.0060	(158)
cplex-333ms	0.0000	(489)	0.0000	(393)	0.9960	(497)	0.9716	(354)	0.0000	(487)	0.3935	(444)
cplex-1000ms	0.0000	(1052)	0.0000	(1039)	0.9960	(1143)	0.9716	(1140)	0.0000	(2887)	0.3935	(1452)
cplex-3000ms	0.0000	(9171)	0.9338	(3563)	0.9964	(3030)	0.9716	(3077)	0.0000	(3090)	0.5804	(4386)

(each value in () is time in milliseconds)

We prepared cut-off results for Casanova and HC. For example, *casanova-10ms* denotes the result of Casanova within 10 milliseconds. Here, for faster approximation, we used *greedy-3*, *GHC-3*, and *BHC-3* but did not use *greedy-11*, *GHC-11*, and *BHC-11*. Here, *greedy-3* uses the best results of Lehmann's greedy allocation algorithm with parameter ($0 \leq c \leq 1$ in 0.5 steps). *GHC-3* and *BHC-3* use the best results of the local updating with parameter ($0 \leq c \leq 1$ in 0.5 steps). Also, we prepared a variant of our algorithm that has a suffix of '-seq' or '-para'. The suffix '-seq' denotes the algorithm is completely executed in a sequence that is equal to one that can be executed on a single CPU computer. For example, *greedy-3-seq* denotes that the execution time is just the sum of execution times of three threads. The suffix '-para' denotes the algorithm is completely executed in a parallel manner, and the three independent threads are completely executed in parallel. Here, we used the ideal value for '-para' since our computer has only two cores in the CPU. The actual execution performance will be between '-seq' and '-para'. Also, we denote the initial performance of Zurel's algorithm as *Zurel-1st*. Here, *Zurel-1st* is the result at the end of its first phase and no winners will be approximately assigned before it. *cplex* is the result of CPLEX with the specified time limit.

On most distributions in Table 2, *Zurel-1st* takes more than 1 second but the obtained *ratioA* is lower than *greedy-3-seq*. Furthermore, the average *ratioA* of *GHC-3-para-1000ms* is higher than *Zurel* while its computation time is less than both *Zurel* and *Zurel-1st*.

On all distributions in Table 2, *BHC* could not get any update within the time limit so there is no update from *greedy*. Here, although *SA* performs better than *greedy(C=0.5)*, it could not outperform *GHC(C=0.5)* in any case. Therefore, we can see that both 'best-updating' and 'random-updating' approaches are not sufficient enough for extremely short time approximation, although the 'greedy-updating' approach makes a good performance in the same situation.

Table 3. Time Performance on 100,000 bids-256 items

	L2	L3	L4	L6	L7	average
greedy-3	1.1098 (51.5)	0.9836 (54.4)	1.0003 (56.8)	1.0009 (58.8)	0.8688 (52.5)	0.9927 (54.8)
GHC-3-para-333ms	1.1098 (333)	0.9859 (333)	1.0003 (333)	1.0009 (333)	0.9395 (333)	1.0073 (333)
GHC-3-para-1000ms	1.1098 (1000)	0.9880 (1000)	1.0003 (1000)	1.0010 (1000)	0.9814 (1000)	1.0161 (1000)
Zurel-1st	0.8971 (74943)	0.9827 (2257)	0.9998 (5345)	0.9987 (4707)	0.7086 (8688)	0.9174 (19188)
Zurel	1.0000 (91100)	1.0000 (6036)	1.0000 (30568)	1.0000 (44255)	1.0000 (17691)	1.0000 (37930)
casanova-130ms	0.3031 (130)	0.0061 (130)	0.0117 (130)	0.0182 (130)	0.2246 (130)	0.1127 (130)
casanova-333ms	0.3506 (333)	0.0379 (333)	0.0328 (333)	0.0673 (333)	0.7536 (333)	0.2484 (333)
casanova-1000ms	0.4954 (1000)	0.1176 (1000)	0.0946 (1000)	0.1605 (1000)	0.7832 (1000)	0.3303 (1000)
cplex-100ms	0.0000 (2022)	0.0000 (232)	0.0000 (143)	0.0000 (133)	0.0000 (852)	0.0000 (676)
cplex-333ms	0.0000 (2021)	0.0000 (559)	0.9998 (1084)	0.0000 (412)	0.0000 (852)	0.2000 (986)
cplex-1000ms	0.0000 (2021)	0.0000 (1045)	0.9998 (1085)	0.0000 (1328)	0.0000 (1285)	0.2000 (1353)
cplex-3000ms	0.0000 (3496)	0.0000 (3286)	0.9998 (5207)	0.9965 (3092)	0.0000 (15667)	0.3993 (6149)

(each value in () is time in milliseconds)

In many settings of CPLEX, the values are 0. This is because CPLEX could not generate initial approximation result within the provided time limit. Only L4 and L6 have results for CPLEX. However, CPLEX spends around 400 msec for the computation but the results are still lower than greedy-3. For L3, CPLEX could prepare results in 3.8 sec of computation, however, the result is still lower than greedy-3. This is because the condition we set up gave extremely short time limit so therefore CPLEX could not generate sufficient approximation results in such hard time constraint.

Table 3 shows the experimental result on the dataset with 100,000 bids in an auction focused on the early anytime performance. While GHC-3 and Zurel’s algorithm are competitive in Table 2, it is clear that our proposed GHC-3 outperforms Zurel’s algorithm in any time performance in Table 3. Note that the time needed to attain initial allocations increased dramatically (approx. 2 times in L3 to over 7 times in L7) when the number of bids becomes five times larger than that of Table 2. However, our GHC-3-para-1000ms only takes the same execution time (i.e, 1000 msec) but its average *ratioA* is higher than Zurel. Note that the GHC-3-para-333ms has still higher *ratioA* value than Zurel while its average computation time is 100 times less. We argue that our algorithm has an advantage when the number of bids increases.

5 Related Work

5.1 Approaches for Optimization Problems

There are really many approaches to optimization problems. Linear programming is one of the well-known approaches in this area. The winner determination problem on combinatorial auctions can be transformed into a linear programming problem. Therefore, it is possible to use a linear programming solver for the winner determination problem.

CPLEX is a well-known, very fast linear programming solver system. In [11], Zurel et al. evaluated the performance of their presented algorithm with many data sets, compared with CPLEX and other existing implementations. While the version of CPLEX used in [11] is not up-to-date, the shown performance of Zurel’s algorithm is approximately 10 to 100 times faster than CPLEX. In this

paper, we showed direct comparisons to the latest version of CPLEX we could prepare. Our approach is far better than latest version of CPLEX for large-scale winner determination problems. Therefore, the performance of our approach is competitive enough with CPLEX or other similar solver systems. This is natural since Zurel's and our approaches are specialized for combinatorial auctions, and also focus only on faster approximation but do not seek optimal solutions. In case we need optimal solutions, it is good choice to solve the same problem by both our approach and CPLEX in parallel. This could improve anytime performance but guarantee obtaining optimal solutions. In this case, our approach should spend very small computation overhead.

Random-walk search is also a strong approach for approximating combinatorial optimization problems. There have been many algorithms proposed based on random-walk search mechanisms.

In [10], Casanova was proposed, which applies a random walk SAT approach for approximating the winner determination problem in combinatorial auctions. In this paper, we showed that our approach outperforms Casanova when the time constraint is very hard but the problem space is really large.

Simulated Annealing (SA) is another similar approach. We prepared an SA-based extension for our approach and we confirmed it increases the performance when the problem size is relatively small. However, SA needs random-walk in the early stage of its search and it decreases performance on short-time approximation.

Genetic Algorithm is another similar approach. In [14], Avasarala et al. proposed an approach for the winner determination problem on combinatorial auctions. However, in [14], they noticed that their algorithm is not effective for approximation in short time but is effective for obtaining higher optimal solutions with enough computation time. Random-walk searching is really effective approximation approach for combinatorial optimization problems. However, it is not effective when there are such hard time constraints. We focused on solving problems that are hard for such random-walk search approaches.

5.2 Approaches to Obtain Optimal Solutions

There have been a lot of works on obtaining optimal solutions for winner determination in combinatorial auctions [12]. For example, CABOB [2] and CASS [15] have been proposed by aiming to get the optimal allocations.

In [10], it is shown that the Casanova algorithm outperforms approximation performance of CASS on winner determination. In this paper, we showed that our approach outperforms Casanova in settings of a very large number of bids in an auction. Therefore, our approach should also outperform CASS in the same settings.

In [2], Sandholm et al. showed that CABOB outperforms CPLEX in several settings. However, according to our comparison, our algorithm should outperform CABOB in our settings. We argue that our approach is rather complementary to those algorithms that are seeking exact optimal solutions. It is not fair to compare their approximation performances when one guarantees obtaining optimal

solutions but the other does not. Our approximation approach only covers large size problem settings that can only be handled by specialized approximation algorithms. Our approach does not contribute to advances in developing algorithms to obtain optimal solutions directly.

5.3 Greedy Approaches

Some researchers have noticed the better performance of simple greedy and incremental approaches for very large-scale problems. For example, [16] noticed the ease of approximation on very large auction problems. In [9], Lehmann et al. mentioned that a simple greedy approach obtains very high results when the auction problem is rather huge.

Also in [17], Kastner et al. mentioned a potential capability of a simple incremental search approach to apply to very large auction problems and discussed the sensitivity for the number of bids in an auction. However, there is little mentioned about a detailed comparison of actual performances for several different types of datasets. In [17], they only presented their preliminary experimental results on a dataset that is based on a single bid distribution.

Guo et al. [18] proposed similar local-search based algorithms and they argued that their approach is good for the settings of a large number of bids in a combinatorial auction problem. However, in [18], they presented very limited experimental results and little analysis or comparison to other high performance algorithms. Also in [18], they did not propose an idea that is similar to our multiple bid-weighting search. We argue that this multiple weighting search approach is very effective and that it distinguishes our approach from others. Also, we showed a detailed analysis of our experiments based on datasets generated by possible different bid distributions. We also showed direct comparisons to Zurel's approach presented in [11].

5.4 Other Approaches

When we have some assumptions about models for valuation of bids, we can utilize those assumptions for better approximation. Dobzinski et al. proposed improved approximation algorithms for auctions with submodular bidders [19]. Lavi et al. reported an LP-based algorithm that can be extended to support the classic VCG [20]. Those studies mainly focused on theoretical aspects. In contrast to those papers, we rather focus on experimental analysis and implementation issues. Those papers did not present experimental analysis of the settings with a large number of bids as we presented in this paper.

Using sequential auctions [21] is another approach to overcoming the communication cost problem. Koenig et al. proposed a multiple-round auction mechanism that guarantees the upper bound of communication cost as fixed size k , that is independent from the number of agents or items in the auction [22]. Although our algorithm itself can approximate winners within a very short time with a huge number of updated bids, the communication cost problem remains.

6 Conclusions

In this paper, we analyze the performance our winner approximation algorithm with slightly modified variants on numerous datasets. We confirmed our parallel greedy approach performs well when the problem is difficult to be solved by an approach with single bid weighting strategy (e.g., $c = 0.5$). Also we showed that our parallel greedy improvement approach performs well when the computation should be completed in very short time. Furthermore, our algorithm works better than CPLEX when the problem is large but its computation should be completed in short time. Especially, our GHC algorithm is effective to solve the issue of ‘no solution’, that is when other anytime algorithms could take a moment to produce an initial solution for the problem.

References

1. Cramton, P., Shoham, Y., Steinberg, R.: Combinatorial Auctions. MIT Press, Cambridge (2006)
2. Sandholm, T., Suri, S., Gilpin, A., Levine, D.: Cabob: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science* 51(3), 374–390 (2005)
3. McMillan, J.: Selling spectrum rights. *The Journal of Economic Perspectives* (1994)
4. Fukuta, N., Ito, T.: Towards better approximation of winner determination for combinatorial auctions with large number of bids. In: Proc. of The 2006 WIC/IEEE/ACM International Conference on Intelligent Agent Technology (IAT 2006), pp. 618–621 (2006)
5. Fukuta, N., Ito, T.: Short-time approximation on combinatorial auctions – a comparison on approximated winner determination algorithms. In: Proc. of The 3rd International Workshop on Data Engineering Issues in E-Commerce and Services (DEECS 2007), pp. 42–55 (2007)
6. Fukuta, N., Ito, T.: Periodical resource allocation using approximated combinatorial auctions. In: Proc. of The 2007 WIC/IEEE/ACM International Conference on Intelligent Agent Technology (IAT 2007), pp. 434–441 (2007)
7. Fukuta, N., Ito, T.: Fast partial reallocation in combinatorial auctions for iterative resource allocation. In: Proc. of 10th Pacific Rim International Workshop on Multi-Agents (PRIMA 2007), pp. 196–207 (2007)
8. Fukuta, N., Ito, T.: Toward a large scale e-market: A greedy and local search based winner determination. In: Okuno, H.G., Ali, M. (eds.) IEA/AIE 2007. LNCS, vol. 4570, pp. 354–363. Springer, Heidelberg (2007)
9. Lehmann, D., O’Callaghan, L.I., Shoham, Y.: Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM* 49, 577–602 (2002)
10. Hoos, H.H., Boutilier, C.: Solving combinatorial auctions using stochastic local search. In: Proc. of the AAAI 2000, pp. 22–29 (2000)
11. Zurel, E., Nisan, N.: An efficient approximate allocation algorithm for combinatorial auctions. In: Proc. of the Third ACM Conference on Electronic Commerce (EC 2001), pp. 125–136 (2001)
12. de Vries, S., Vohra, R.V.: Combinatorial auctions: A survey. *International Transactions in Operational Research* 15(3), 284–309 (2003)
13. Leyton-Brown, K., Pearson, M., Shoham, Y.: Towards a universal test suite for combinatorial auction algorithms. In: Proc. of EC 2000, pp. 66–76 (2000)

14. Avasarala, V., Polavarapu, H., Mullen, T.: An approximate algorithm for resource allocation using combinatorial auctions. In: Proc. of the 2006 WIC/IEEE/ACM International Conference on Intelligent Agent Technology (IAT 2006), pp. 571–578 (2006)
15. Fujishima, Y., Leyton-Brown, K., Shoham, Y.: Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In: Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999), pp. 548–553 (1999)
16. Sandholm, T.: Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence* 135, 1–54 (2002)
17. Kastner, R., Hsieh, C., Potkonjak, M., Sarrafzadeh, M.: On the sensitivity of incremental algorithms for combinatorial auctions. In: Proc. International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002), pp. 81–88 (2002)
18. Guo, Y., Lim, A., Rodrigues, B., Zhu, Y.: A non-exact approach and experiment studies on the combinatorial auction problem. In: Proc. of HICSS 2005, p. 82.1 (2005)
19. Dobzinski, S., Schapira, M.: An improved approximation algorithm for combinatorial auctions with submodular bidders. In: SODA 2006: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, pp. 1064–1073. ACM Press, New York (2006)
20. Lavi, R., Swamy, C.: Truthful and near-optimal mechanism design via linear programming. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), pp. 595–604 (2005)
21. Boutilier, C., Goldszmidt, M., Sabata, B.: Sequential auctions for the allocation of resources with complementarities. In: Proc. of International Joint Conference on Artificial Intelligence (IJCAI 1999), pp. 527–534 (1999)
22. Koenig, S., Tovey, C., Zheng, X., Sungur, I.: Sequential bundle-bid single-sale auction algorithms for decentralized control. In: Proc. of International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 1359–1365 (2007)

Online Market Coordination

Masabumi Furuhata^{1,2}, Dongmo Zhang¹, and Laurent Perrussel²

¹ Intelligent Systems Laboratory, University of Western Sydney, Australia

² IRIT-Université de Toulouse, France

Abstract. The aim of this paper is to deal with the problem of coordination for online markets where a seller registers to an online market to sell an item. The seller and the owner of the market then form an alliance to generate revenue through online sales. However, the efficiency and stability of the alliance highly relies on the contract that specifies the way to split the revenue and costs over the alliance members. We consider some typical contracts and examine their influences on the behavior of the alliance. We introduce the key concept of alliance coordination which characterizes the efficiency and stability of an online market.

1 Introduction

The explosive growth of online markets (or e-markets) has caused many changes in the way business is done traditionally. An online market is a web-based facility with which multiple traders can sell or buy goods and services through the Internet. Typical online markets are eBay, Amazon, lastminute.com, and so on. Different from the traditional markets, the partnership between traders and the market owner in an online market can be loosely tied and dynamic in most situations. A trader can enter the market any time and could leave the market any time also even during a transaction. The market owner has poor information about the traders. In addition, an online market can normally accommodate thousands of traders to trade in the market. Monitoring the behavior of each trader is hard. Therefore the mechanism that coordinates the market owner and traders in an online market is critical to the efficiency, effectiveness and stability of the market.

We consider a typical situation whereby a seller registers to an online market to sell a certain product. The seller and the owner of the market then form an alliance of business to generate revenue through online sales. However, the efficiency and stability of the alliance relies on the mechanism (contract) that specifies the way to split the revenue and costs over the members of alliance. Thus we are interested in alliance coordinating contract that gives each party positive expected profit and there is no other contract which gives a better profit to one party without sacrificing the other party's profit. In other words, there is no better contract such that both parties of the alliance are happy to move to. Therefore the concept of coordination specifies the efficiency and stability of an alliance.

We will consider some typical contracts and examine whether they coordinate an online market alliance. One of the simplest and widely used contract in e-business is fixed-fee charging, i.e. the market owner always gets a fixed amount from the revenue regardless the amount of overall revenue and costs of the alliance. We prove that if the costs of online market can be ignored, fixed-fee charging coordinates the alliance.

However, the result is no longer true if the costs of market owner are significant. This result includes an important aspect of the online market. A typical example is to put an advertisement works to increase revenue. Unlike the traditional supply chains, the cost owner of advertisement can be the online market owner instead of the sellers. We show that the property of contracts may change according to the effect of the costs using an advertisement example. Next, we consider another common used contract – revenue sharing, under which each party receives a certain percentage of revenue from the overall revenue. Unfortunately, the contract is unable to coordinate an alliance even if the costs of the market owner are ignorable. Finally we represent *profit sharing contract* that coordinates the online market alliance. Profit sharing contract predetermine a proportion of sharing revenue among the alliance members. Meanwhile, the cost is shared by the opponent’s proportion of the revenue. Hence it balances out the revenue and the cost among the alliance members.

As far as we know, few works have tackled this problem in online market. In [1], Netessine et al. consider the problem of coordination for online retailers that behave similar to the traditional supply chains [2][3]. That is, online retailers buy products and sell them online. In our model, we do not assume this behavior (the online market owner do not buy these products). As a result, our proposal differs from the coordination in the traditional supply chains.

In Section 2, we present the concepts of contract and alliance coordination based on a generic market model. In Section 3, we detail the characteristics of online market. In the next two sections, we formally characterize the properties of fixed-fee contract (Section 4) and revenue-sharing contract (Section 5) in the context of online market. In Section 6, we propose a contract that achieves alliance coordination for online market. Finally, in Section 7, we discuss some related works and conclude the paper.

2 Alliance Coordinating Contracts

As soon as agents commit to a joint action in order to obtain some revenues they have to define the way they will share these revenues. This way is usually detailed in a contract that leads agents to establish an alliance. The aim of a contract is to detail the revenue for each member of the alliance. Knowing the contract, members of the alliance will determine their actions. In the following, we formally describe all these basic concepts.

2.1 Contract in Alliance

Consider an alliance \mathcal{A} in which there are n agents. Each agent i ($1 \leq i \leq n$) has a strategy space S_i . Let $\mathcal{S} = \prod_{i=1}^n S_i$. Each agent i chooses a strategy $\sigma_i \in S_i$. Let $\sigma = (\sigma_1, \dots, \sigma_n) \in \mathcal{S}$ be a strategy profile and σ_{-i} be the strategy profile for all agents except i . We assume that alliance \mathcal{A} earns an alliance revenue as the result of the strategic choices of all agents in \mathcal{A} . Let $R : \mathcal{S} \rightarrow \mathbb{R}$ be a function representing alliance revenue. In many cases, the strategy is interpreted as investments, efforts, or actions incurring costs to generate revenue. Since the alliance revenue is generated as a result of joint-work among the agent, the way to share the revenue is significant for

the alliance members. In general, the way of sharing is specified in a contract defined as follows.

Definition 1. A contract of an alliance \mathcal{A} is a function $\tau : \mathcal{S} \times \mathfrak{R} \rightarrow \mathfrak{R}^n$ which satisfies $\sum_{i \in \mathcal{A}} \tau_i(\sigma, r) = r$ for any $\sigma \in \mathcal{S}$ and $r = R(\sigma)$.

That is, w.r.t. an alliance revenue, a contract defines individual revenues for each agent based on the strategic choices. We assume the alliance revenue is non-negative. Each individual revenue may include fees and incentives, hence it may be negative.

Example 1. Consider a joint investment alliance on natural resource development consisting of three investors, $\mathcal{A} = \{a, b, c\}$. Let σ be a set of monetary investments of all alliance members. According to the investments, natural resource $r = R(\sigma)$ is mined. Suppose all members agree on the following contract before the investments: Each share of r is proportional to the individual amount of investment; that is $\tau_i(\sigma, r) = r \left(\frac{\sigma_i}{\sum_{i \in \mathcal{A}} \sigma_i} \right)$ for all i .

The above example is an ideal case, since the profits are shared among the alliance members. However, the profit-sharing is not always effective or implementable in industry. For instance, an individual cost cannot always be associated to specific alliance revenue or cost information is usually private information.

Definition 1 shows that individual revenues are based on strategies and contracts. It means that agents may choose their strategies with respect to the contract. In order to choose the strategies agents need criteria to evaluate their profits in terms of strategies and contracts. Let $\pi_i^\tau(\sigma)$ be the profit of agent i at strategy profile σ and contract τ . Profit is based on revenue and cost. Let $c_i(\sigma_i)$ be the cost function of agent i . Then the profit function is $\pi_i^\tau(\sigma) = \tau_i(\sigma, r) - c_i(\sigma_i)$. We assume that agents choose their own strategies to maximize their own profits. Since the profit of agent i depends on the other agents' strategies, the evaluation of profits is explained by Nash equilibrium. Formally:

Definition 2. Given alliance \mathcal{A} , contract τ , the set of all strategy spaces \mathcal{S} , and profits $\{\pi_i^\tau(\sigma)\}_{i \in \mathcal{A}}$, the profile $\hat{\sigma} \equiv (\hat{\sigma}_1, \dots, \hat{\sigma}_n)$ is called a Nash equilibrium if for all $i \in \mathcal{A}$, and for all $\sigma_i \in S_i$, $\pi_i^\tau(\hat{\sigma}) \geq \pi_i^\tau(\sigma_i, \hat{\sigma}_{-i})$.

It means that at Nash equilibria, no agent can increase its profit by changing strategies w.r.t. contract τ . However, even if we have a Nash equilibrium, the overall profit may not be maximized. Alliance optimal profit is gained by Pareto optimal contract as follows.

Definition 3. A contract τ is Pareto optimal if there is no other contract τ' such that for any strategy profile $\sigma \in \mathcal{S}$ such that $\pi_i^{\tau'}(\sigma) \geq \pi_i^\tau(\sigma)$ for all i with at least one strict inequality.

It means that if a contract is Pareto optimal, then there is no other contract where all the agents' profits can increase. Notice that our definition of Pareto optimal contract avoids the case where increasing one agent's profit is possible without decreasing others agents' profits. To investigate Pareto optimal contracts, we have to compare profits for each strategy profile, each contract and each agent. In order to simplify this task, we

introduce the notion of alliance optimal profit. Let the alliance profit be $\pi(\sigma) = R(\sigma) - C(\sigma)$, where $C(\sigma)$ is a linear combination of each cost $c_i(\sigma_i)$. Let the alliance optimal profile be $\sigma^* \equiv (\sigma_1^*, \dots, \sigma_n^*)$ such that $\sigma^* = \arg \max_{\sigma \in S} \pi(\sigma)$. The following proposition shows how to check Pareto optimality of contracts.

Proposition 1. *Let σ^* be an alliance optimal strategy profile and let $\hat{\sigma}^\tau$ be the unique Nash equilibrium under a contract τ . If $\hat{\sigma}^\tau = \sigma^*$, then contract τ is a Pareto optimal.*

Proof. The assumptions of the proposition entail that we have $\sum_{i \in \mathcal{A}} \pi_i^\tau(\hat{\sigma}^\tau) = R(\hat{\sigma}^\tau) - \sum_{i \in \mathcal{A}} c_i(\hat{\sigma}^\tau) = R(\sigma^*) - C(\sigma^*) = \pi(\sigma^*)$. Since a strategy profile σ^* maximizes the alliance profit, there is no other way to increase the alliance profit. Hence, contract τ is Pareto optimal. \square

Choosing contract and setting its parameters is a way to give incentive to agents for their participation in an alliance. For instance, contract may guarantee some revenue. This is especially significant if demand has to be considered as uncertain. Taking into account uncertainty of demand is mandatory, since agents have to decide their behavior before they realize the actual demand. Decision criteria of agents are dependent on risk tendencies. Since we assume all the agents are risk neutral, these decisions are only dependent on the expected profits. We denote E as expectation for stochastic variables. Based on the above settings, we define an alliance coordinating contract as follows.

Definition 4. *Given alliance \mathcal{A} , contract τ coordinates alliance \mathcal{A} , if it satisfies the following conditions,*

1. *contract τ is Pareto optimal,*
2. *there exists a strategy profile σ such that $E[\pi_i^\tau(\sigma)] > 0$ for all $i \in \mathcal{A}$.*

The definition shows that our interested contracts must be acceptable by every agents. That is, in addition to the Pareto optimality, the alliance coordination requires that all expected profits should be positive; this constraint is called participation constraint.

3 Online Market Model

In the previous section, we have shown the definition of contract and the alliance coordinating contract. Let us instantiate this framework in the context of online market. We consider an online market where the sellers and buyers trade items. In this online market, an alliance consists of an online market owner and a seller.

For strategic choices of the two alliance members, we consider the seller chooses the listing quantity of a single type of items on the online market and the online market owner chooses advertisement amount. As mentioned, we suppose that the sales depend on a given stochastic demand at given price. The revenue is shared between the two alliance members according to the contract. The seller directly ships the item to the buyers and thus buyers are not considered as members of the alliance.

Formally, let $\mathcal{A} = \{o, s\}$ be the alliance such that o is the online market owner and s is the seller. The strategy of o is to choose advertisement amount a and the strategy of s is to choose listing quantity q . For a given price p , we assume a random demand $X(a)$ which is affected by advertisement amount a . Let F be the cumulative

distribution function of the demand and f be its probability distribution function. If the listing quantity is q and the advertisement amount is a , we obtain the expected revenue $R(q, a) = pQ(q, a)$ where $Q(q, a)$ is the expected sales quantity such that $Q(q, a) = E[\min\{X(a), q\}]$. It means that if listing quantity q is greater than demand, then sales quantity is a demand, otherwise the sale quantity is inventory quantity q . Stochastic demand entails that $E[\min\{X(a), q\}]$ and thus $Q(q, a)$ are equal to:

$$Q(q, a) = \int_0^q xf(x|a)dx + \int_q^\infty qf(x|a)dx. \tag{1}$$

By differentiating $Q(q, a)$ w.r.t. q , we get that the increase of inventory quantity for one unit results in the increase of the expected sales quantity less than one unit (since $\frac{\partial Q(q,a)}{\partial q} = 1 - F(q|a) < 1$). For the advertisement effect, we assume positive effect on the expected sales (that is $\frac{\partial Q(q,a)}{\partial a} \geq 0$). Furthermore, we assume that the expected sales is diminishing concave (i.e. $\frac{\partial^2 Q(q,a)}{\partial a^2} \leq 0$ and $\frac{\partial^3 Q(q,a)}{\partial a^3} \geq 0$).

For online trades, we consider the following costs for online market owner o and seller s . For agent o the cost is equal to the fixed cost $c_{\bar{o}}$ plus advertisement cost. Let $g(a)$ be a cost function for advertisement a where $g(0) = 0$, $\frac{dg(a)}{da} > 0$ and $\frac{d^2g(a)}{da^2} \geq 0$. In other words, the cost of advertisement and the marginal advertisement cost both increase w.r.t. advertisement amount. For seller s , at the time of listing quantity on the online market, we assume that items are already prepared as inventories with unit cost c_p . We also consider fixed cost for seller s as $c_{\bar{s}}$ and shipment cost c_s per unit. For simplicity, we do not consider either salvage value, stock out penalty, or inventory holding cost.

At the opposite of listing quantity, advertisement can be null to generate revenue. In such a case, the alliance strategy (q, a) is equal to $(q, 0)$ and whenever it is clear q stands for the strategy. Let us detail the case where there is no advertisement. By choosing listing quantity q , the alliance expects to earn revenue $r = R(q) = pQ(q)$ at given price p . Suppose the alliance agrees on a contract τ , the expected profit of online market owner o is

$$E[\pi_o^\tau] = \tau_o(q, r) - c_{\bar{o}} \tag{2}$$

and the expected profit of seller s is

$$E[\pi_s^\tau] = \tau_s(q, r) - c_sQ(q) - c_pq - c_{\bar{s}} \tag{3}$$

These two functions show that the profits of alliance members depend on the choice of listing quantity by the seller and of the contract. Meanwhile, the choice of the listing quantity incurs variable costs for the seller however it does not incur any variable costs for agent o . This setting is very unique for online market compared to the traditional supply chains [2]. While physical distributions are executed among alliance members that incur variable costs in the traditional supply chains, online market does not incur any variable costs due to direct shipment from the seller to the buyer.

Based on the above settings, we first show that there exists an alliance optimal profit for this model.

Lemma 1. *Let q^* be an alliance optimal listing quantity in the online market model without advertisement effect. There exists a unique optimal listing quantity $q^* = F^{-1}\left(\frac{p-c_s-c_p}{p-c_s}\right)$ in the online market model.*

Proof. According to the definition of the profit, we obtain the alliance expected profit as follows, $E[\pi(q)] = (p - c_s)Q(q) - c_pq - (c_{\bar{o}} + c_{\bar{s}})$. The first-order derivative of the expected profit is

$$\frac{dE[\pi(q)]}{dq} = (p - c_s)(1 - F(q)) - c_p \tag{4}$$

The second-order derivative of Equation (4) is $\frac{d^2E[\pi(q)]}{dq^2} = -(p - c_s)f(q)$. Since $f(q)$ is positive, we obtain that the alliance profit function is concave in quantity q . Therefore, the alliance optimal quantity q^* must be the solution of Equation (4) such that $q^* = F^{-1}\left(\frac{p-c_s-c_p}{p-c_s}\right)$. \square

Lemma 1 shows that the unique alliance optimal quantity exists in this model. Therefore, an equilibrium listing quantity under a certain contract must be equal to q^* . Based on this optimal quantity, we investigate the alliance coordinating contracts in the case of no advertisement. We focus on an interesting case where $E[\pi(q^*)] > 0$.

Now let us relax the assumption of no advertisement. Taking into account advertisement entails to redefine the Pareto optimality checking, since the alliance profit function is different. The alliance expects to earn revenue $r = R(q, a) = pQ(q, a)$ at given price p . Suppose the alliance agrees on a contract τ , the expected profit of agent o is

$$E[\pi_o^\tau(q, a)] = \tau_o(q, a, r) - c_{\bar{o}} - g(a), \tag{5}$$

and the expected profit of agent s is

$$E[\pi_s^\tau(q, a)] = \tau_s(q, a, r) - (1 - \alpha)c_sQ(q, a) - c_pq - c_{\bar{s}} \tag{6}$$

In order to check the alliance coordination, we define the expected alliance profit as follows,

$$E[\pi(q, a)] = (p - c_s)Q(q, a) - c_pq - (c_{\bar{o}} + c_{\bar{s}}) - g(a). \tag{7}$$

According to the definitions of Q , Equation (7) is concave in listing quantity q and a . Therefore, there exists an alliance optimal pair $\{q^*, a^*\}$. Notice that this pair is not necessary unique. We suppose that for any given fixed listing quantity q , there exists optimal advertisement amount $a^*(q)$. Formally, this is represented by the following first-order condition, similarly to [43]:

$$\frac{\partial \pi(q, a^*(q))}{\partial a} = (p - c_s) \frac{\partial Q(q, a^*(q))}{\partial a} - \frac{dg(a^*(q))}{da} = 0 \tag{8}$$

For this online market model, we now investigate the properties of alliance coordinating contracts. We focus on two typical contracts: *fixed-fee contract* and *revenue-sharing contract*. For these two contracts, we study the advertisement effect.

4 Fixed-Fee Contract

Fixed-fee contract is employed in many online markets. Fixed-fee contract is a contract where one agent always gets the same individual revenue regardless the alliance revenue, formally:

Definition 5. A contract τ of an alliance \mathcal{A} is called to be fixed-fee contract by agent i_0 if it satisfies the following conditions: for any $\sigma \in \mathcal{S}$ and $r \in \mathfrak{R}$,

1. $\tau_{i_0}(\sigma, r) = \alpha$
2. $\sum_{i \neq i_0} \tau_i(\sigma, r) = r - \alpha$

where $\alpha \in \mathfrak{R}$ is constant and interpreted as the charging fee.

Under fixed-fee contract, agent i_0 charges fixed-fee α to the other agents and the returns of the alliance is taken by agents except for agent i_0 . In the context of online market, we have $\tau_o(\sigma, r) = \alpha$ and $\tau_s(\sigma, r) = r - \alpha$ s.t. $\sigma = (q, a)$. Notice that charging a membership fee is a similar contract.

Example 2. Consider an online market where sellers sell second-hand items to buyers. In this online market, the owner o charges \$2 fixed-fee to seller s for each listing. The contract can be represented as follows:

$$\tau_o(\sigma, r) = 2; \quad \tau_s(\sigma, r) = r - 2.$$

It means for any seller's strategy the market owner's share of revenue is constant. This contract is used in eBay BuyItNow option.

4.1 No Advertisement

The following proposition shows that the online market model without advertisement effect achieves alliance coordination.

Proposition 2. Let α be a fixed-listing fee of the online market. Fixed-fee contract τ achieves alliance coordination in online market model without advertisement effect, if $c_{\bar{o}} < \alpha < (p - c_s)Q(q^*) - c_p q^* - c_{\bar{s}}$.

Proof. If $q > 0$, under fixed-fee contract, the expected profit of online market owner o and seller s are respectively,

$$E[\pi_o^\tau(q)] = \alpha - c_{\bar{o}} \tag{9}$$

$$\begin{aligned} E[\pi_s^\tau(q)] &= r - \alpha - c_s Q(q) - c_p q - c_{\bar{s}} \\ &= (p - c_s)Q(q) - c_p q - \alpha - c_{\bar{s}}, \end{aligned} \tag{10}$$

otherwise, we have $\pi_o^\tau(q) = \pi_s^\tau(q) = 0$. Since the expected profit of online market owner o is always α if $q > 0$, the online market owner concerns whether the seller lists items at quantity $q > 0$ or not. Hence, we check the optimal listing quantity for seller s denoted as \hat{q}_s . By differentiating the profit function of seller s w.r.t. q , we obtain $\frac{dE[\pi_s^\tau(q)]}{dq} = (p - c_s)(1 - F(q)) - c_p$. According to Equation (4), we get $\frac{dE[\pi_s^\tau(q)]}{dq} =$

$\frac{dE[\pi(q)]}{dq}$ and according to Lemma 1 there is a unique alliance optimal quantity q^* in this model. Thus we obtain $\hat{q}_s = q^*$. For participation constraints, Equation (9) and (10) must be positive at $q = q^*$. Therefore, fixed-fee contract achieves alliance coordination, if fixed-fee α satisfies $c_{\bar{o}} < \alpha < (p - c_s)Q(q^*) - c_p q^* - c_{\bar{s}}$. \square

Proposition 2 shows that the alliance coordination is due to the cost structure of the online market owner which does not incur variable cost for the listing quantity. As long as the fee is greater than the owner's cost and lower than the seller's expected profit, the alliance coordination holds.

4.2 Advertisement Effect

According to the advertisement effect, strategies are now pairs (q, a) and the contracts are $\tau_o(q, a, r) = \alpha$ and $\tau_s(q, a, r) = r - \alpha$. The following proposition shows that fixed-fee contract does not achieve alliance coordination.

Proposition 3. *Fixed-fee contract τ does not achieve alliance coordination in the online market model with advertisement effect.*

Proof. Under fixed-fee contract τ , if $q > 0$, according to Equation (5) and the definition of the contract, the profit function of online market owner o is: $E[\pi_o^r(q, a)] = \alpha - c_{\bar{o}} - g(a)$. The first-order derivative of profit function of online market owner o w.r.t. a is $\frac{d\pi_o(a)}{da} = -\frac{dg(a)}{da} < 0$. Hence, under fixed-fee contract, online market owner o does not have incentive to place any positive advertisement amount which is the assumption of the online market model without advertisement effect shown in the previous section. Therefore, fixed-fee contract does not achieve alliance coordination in this model. \square

According to Proposition 3 seller s enjoys a benefit of advertisement effect as a free rider under fixed-fee contract. Furthermore, the online market owner does not have any incentive to place advertisement in a context of alliance coordination.

We have shown that fixed-fee contract achieves alliance coordination in the limited case where advertisement is not considered. The next question we address is whether the other popular contract, revenue-sharing contract, achieve alliance coordination.

5 Revenue-Sharing Contract

The following contract, *revenue-sharing contract*, is frequently used in the online markets. Individual revenue is a proportion of the alliance revenue [3].

Definition 6. *A contract τ of an alliance \mathcal{A} is called to be revenue-sharing contract if there exists $\alpha_1, \dots, \alpha_n$ s.t. $\sum_{i \in \mathcal{I}} \alpha_i = 1$ and for any $(\sigma_1, \dots, \sigma_n) \in \mathcal{S}$ and $r = R(\sigma)$,*

$$\tau_i(\sigma_1, \dots, \sigma_n, r) = \alpha_i r \text{ for all } i$$

Example 3. Consider an online music market for selling songs. The alliance consists of online music store o and music label s . The contract specifies the following royalties on revenue r : 20% of r for agent o and 80% for agent s . Hence, the contracts are:

$$\tau_o(\sigma, r) = 0.20r; \quad \tau_s(\sigma, r) = 0.80r$$

5.1 No Advertisement Effect

Under revenue-sharing contract, the online market owner charges a certain portion of the sales amount of the seller. Portion α ranges in $0 < \alpha < 1$. Hence $\tau_o(q, r) = \alpha r$ and $\tau_s(q, r) = (1 - \alpha)r$. The following proposition shows that this contract does achieve alliance coordination.

Proposition 4. *Let α be a portion that online market owner o earns from the revenue r . Revenue-Sharing contract τ does not achieve alliance coordination in the online market model without advertisement effect.*

Proof. Under revenue-sharing contract, the expected profit of online market owner o is

$$\begin{aligned} E[\pi_o^\tau(q)] &= \tau_o(q, r) - c_{\bar{o}} \\ &= \alpha pQ(q) - c_{\bar{o}} \end{aligned}$$

and the expected profit of seller s is

$$\begin{aligned} E[\pi_s^\tau(q)] &= \tau_s(q, r) - c_sQ(q) - c_pq - c_{\bar{s}} \\ &= (1 - \alpha)pQ(q) - c_sQ(q) - c_pq - c_{\bar{s}}. \end{aligned}$$

The first-order condition for the profit maximizing quantity of seller s is $\frac{dE[\pi_s^\tau(q)]}{dq} = ((1 - \alpha)p - c_s)(1 - F(q)) - c_p = 0$. Let \hat{q}_s be the profit maximizing quantity of the seller under revenue-sharing contract. We obtain $\hat{q}_s = F^{-1}\left(\frac{p(1-\alpha) - c_s - c_p}{p(1-\alpha) - c_s}\right) > q^*$. Hence, revenue-sharing contract does not achieve alliance coordination. \square

As mentioned, the online market owner does not incur any variable cost or any procurement cost. Therefore, revenue-sharing contract does not achieve alliance coordination. Even though the proposition shows that revenue-sharing contract is not an alliance coordinating contract, it is a popular contract in online market. Parameter α is usually set at a small value in the online markets. Therefore, it entails that seller s may list slightly greater quantities than the alliance optimal quantity. This means that the listed quantity entailed by a revenue-sharing contract may be greater than the one entailed by fixed-fee contract. Therefore, the online market owner may sell greater quantities under revenue-sharing contract compared to fixed-fee contract.

5.2 Advertisement Effect

Let α be the online market owner o 's portion of revenue. The contracts are $\tau_o(q, a, r) = \alpha r$ and $\tau_s(q, a, r) = (1 - \alpha)r$. Again we show that revenue-sharing contract does not achieve alliance coordination.

Proposition 5. *Revenue-sharing contract τ does not achieve alliance coordination in the online market model with advertisement effect.*

Proof. Under revenue-sharing contract τ , for a given listing quantity q , let $\hat{a}_o(q)$ be the optimal advertisement amount for online market owner o corresponding to listing quantity q . It entails that the first-order condition represented in Equation (8) holds

for $(q, \hat{a}_o(q))$. Hence, for the optimal profit function π_o^τ , we have $\frac{\partial E[\pi_o^\tau(q, \hat{a}_o(q))]}{\partial a} = \alpha(p - c_s) \frac{\partial Q(q, \hat{a}_o)}{\partial a} - \frac{dg(\hat{a}_o)(q)}{da} = 0$. According to Equation (8), if $\hat{a}_o = a^*$ we have $\frac{\partial E[\pi_o^\tau(q, \hat{a}_o)]}{\partial a} < \frac{\partial E[\pi(q, a^*(q))]}{\partial a}$. Thus we have $\hat{a}_o \neq a^*$. Hence, revenue-sharing contract does not achieve alliance coordination in the case of individual advertisement. \square

According to Propositions 4 and 5, revenue-sharing contract does not achieve the alliance coordination regardless of advertisement effect. This is mainly due to the lack of relation between marginal cost and marginal profit. In the next section, we propose a contract that takes care of this relation.

6 Profit Sharing Contract

The aim of this contract is to balance out revenue and variable costs between alliance members. That is each member does not only consider its cost to define its profit, but also the other members' costs. The revenue is $r = R(\sigma_i, \sigma_j)$. Let $\chi_i > 0$ be a parameter for setting at first the portion of revenue for agent i and, second the portion of cost that agent j charges to agent i . We assume that $\sum_{i \in \mathcal{A}} \chi_i = 1$. The following contract is in the scheme of profit sharing contract in [5],

Definition 7. Let $\chi_i > 0$ be a portion parameter of profit sharing contract τ . A contract τ of an alliance \mathcal{A} is a profit sharing contract if $\tau_i(\sigma_i, \sigma_j, r) = \chi_i r - \chi_i c_j(\sigma_j) + \chi_j c_i(\sigma_i)$ for all $i \in \mathcal{A}$.

In the context of online market, profit sharing contract is interpreted as follows:

$$\begin{aligned}\tau_o(q, a, r) &= \chi r - \chi c_s S(q, a) - \chi c_p q + (1 - \chi)g(a) \\ \tau_s(q, a, r) &= (1 - \chi)r + \chi c_s S(q, a) + \chi c_p q - (1 - \chi)g(a)\end{aligned}$$

This profit sharing contract is a combination of revenue-sharing, sales discount, listing incentive and advertisement cost sharing. The following theorem shows that profit sharing contract achieves alliance coordination.

Theorem 1. Profit sharing contract τ achieves alliance coordination in the online market model.

Proof. W.r.t. τ , the profit function for online market owner o is

$$E[\pi_o^\tau(q, a)] = \chi((p - c_s)S(q, a) - c_p q - g(a)) - c_{\bar{o}}, \quad (11)$$

and the profit function of seller s is

$$E[\pi_s^\tau(q, a)] = (1 - \chi)((p - c_s)S(q, a) - c_p q - g(a)) - c_{\bar{s}}. \quad (12)$$

By differentiating Equation (11) w.r.t. a , we obtain a marginal profit of online market owner w.r.t. advertisement $\frac{\partial E[\pi_o^\tau(q, a)]}{\partial a} = \chi \left((p - c_s) \frac{\partial S(q, a)}{\partial a} - \frac{dg(a)}{da} \right)$. Since $(p -$

$c_s) \frac{\partial S(q,a)}{\partial a} - \frac{dg(a)}{da} = \frac{\partial E[\pi(q,a)]}{\partial a}$, we have $\frac{\partial E[\pi_o^\tau(q,a)]}{\partial a} = \chi \frac{\partial E[\pi(q,a)]}{\partial a}$. Therefore, it satisfies the first-order condition shown in Equation (8). A condition to satisfy participation constraint is Equation (11) and (12) must be positive at a pair $\{q^*, a^*\}$ as follows:

$$E[\pi_o^\tau(q^*, a^*)] = \chi ((p - c_s)S(q^*, a^*) - c_p q^* - g(a^*)) - c_{\bar{o}} > 0$$

$$E[\pi_s^\tau(q^*, a^*)] = (1 - \chi) ((p - c_s)S(q^*, a^*) - c_p q^* - g(a^*)) - c_{\bar{s}} > 0$$

Therefore, we obtain that profit sharing contract achieves alliance coordination, if

$$\frac{c_{\bar{o}}}{(p - c_s)S(q^*, a^*) - c_p q^* - g(a^*)} < \chi < \frac{(p - c_s)S(q^*, a^*) - c_p q^* - g(a^*) - c_{\bar{s}}}{(p - c_s)S(q^*, a^*) - c_p q^* - g(a^*)}$$

□

Profit sharing contract charges their costs to the alliance partner with prefixed portion each other. At the same time, based on the opposite portion, the revenue is shared. According to Theorem 1 profit sharing contract balances out the costs of the alliance members by sharing them. So far, we are not aware of any online market companies that implement this contract. This is because to obtain cost information from the seller is costly and to reveal online market’s cost information to the sellers is too sensitive.

7 Conclusion and Related Work

In this paper, we have at first presented the notion of contract and alliance coordination. Next we have shown how this framework can be used to describe a specific kind of market namely online market. Then, we have studied behavior of this market w.r.t two popular contracts: fixed-fee contract and revenue-sharing contract. We have shown that it is difficult to obtain coordination for these two contracts: only fixed-fee contract with no advertisement achieves alliance coordination. Revenue-sharing contract leads sellers to list greater quantities compared to the case of fixed-fee contract; this property may be a desirable one for gaining market shares. We finally exhibit a profit sharing contract that enables to achieve coordination. Even if this contract is difficult to implement, it may be used for setting more general conditions of alliance coordination, since this contract characterizes the aspects of efficiency and stability. For instance, profit sharing can be approximated to revenue-sharing plus fixed-fee in very limited cases. However, profit sharing is able to indicate the revenue-sharing plus fixed-fee contract’s parameter settings. This contract is actually implemented by eBay’s auction.

Our definition of contract slightly differs from the one given by Gan, et al. in [6]. They define a contract as a proportion of the alliance revenue. Their definition mainly focuses on revenue-sharing contract. Our framework is more suitable for describing different types of contract such as fixed-fee contract.

We have assumed that all agents are risk neutral similar to others settings [2,3,1]. Therefore, we consider that the decision making criteria of the agents are their expected profits. If we want to consider risk averse agents, our definition of alliance coordination may be extended for taking it account utilities of agents as proposed in [6,7].

We remark that there is a significant difference between the concepts of coalition formation in game theory and alliance coordination we have discussed in this paper, though

both of them concern about how a group of agents share the gains from cooperation. Coalition games are described in terms of payoffs of coalitions (subgroups) rather than payoffs of individuals [8]. The main concern of a player in a coalition game is which subgroup he/she should join in order to maximize his/her outcomes. In our model, we assume that all agents are in the same alliance, i.e., a grand coalition. The concern of an agent is how much he/she should invest to the coalition to get the maximal return, given a certain coordination contract.

In this model, we did not consider some specific aspects of online markets like non-cooperative shipment which is the most significant problem in eBay like market according to [9]. In order to deal with this problem, it is significant to embed the concept of reputation into this model. We set this point as future work.

In online market, a key question is to know how to attract traders. For the online market owners such as eBay or lastminute.com, it involves uncertain and incomplete information. As a consequence, it is not always possible to find the optimal strategies. In order to find the solutions for this complex problem, Trading Agent Competition in Market Design (TAC-MD) has been proposed as a simulation test-bed [10]. We can view the problems of TAC-MD as a coordination problem and thus as future work we want to show how our proposal fits the TAC-MD framework.

References

1. Netessine, S., Rudi, N.: Supply chain structures on the internet and the role of marketing-operations interaction. In: Simchi-Levi, D., Wu, D., Shen, Z.M. (eds.) *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*, pp. 607–641. Kluwer, Dordrecht (2004)
2. Cachon, G.P.: Supply chain coordination with contracts. In: de Kok, A., Graves, S.C. (eds.) *Supply Chain Management: Design, Coordination and Operation*, pp. 229–340. Elsevier, Amsterdam (2003)
3. Cachon, G., Larivière, M.: Supply chain coordination with revenue-sharing contracts: Strengths and limitations. *Management Science* 51(1), 30–44 (2005)
4. Petruzzi, N.C., Dada, M.: Pricing and the newsvendor problem: A review with extensions. *Operations Research* 47(2), 183–194 (1999)
5. Jeuland, A.P., Shugan, S.M.: Managing channel profits. *Marketing Science* 2(3), 239–272 (1983)
6. Gan, X., Sethi, S.P., Yan, H.: Coordination of supply chains with risk-averse agents. *Production and Operations Management* 13(2), 135–149 (2004)
7. Shum, W.S.: *Effective Contracts in Supply Chains*. PhD thesis, the Sloan School of Management, Massachusetts Institute of Technology (2007)
8. Osborne, M.J., Rubinstein, A.: *A Course in Game Theory*. MIT Press, Cambridge (1994)
9. Resnick, P., Zeckhauser, R.: Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. In: Baye, M.R. (ed.) *The Economics of the Internet and E-Commerce*, vol. 11, pp. 127–157. Elsevier Science, Amsterdam (2002)
10. Niu, J., Cai, K., Parsons, S., Gerding, E., McBurney, P.: Characterizing effective auction mechanisms: Insights from the 2007 tac market design competition. In: *AAMAS*, pp. 1079–1086 (2008)

Towards an Evaluation Framework for MAS Software Engineering

Emilia Garcia, Adriana Giret, and Vicente Botti

Universidad Politecnica de Valencia, Camino de Vera, Valencia, Spain
mgarcia@dsic.upv.es, agiret@dsic.upv.es, vbotti@dsic.upv.es

Abstract. Recently a great number of methods and frameworks to develop multiagent systems have appeared. It makes difficult the selection between one and another. Because of that the evaluation of multiagent system software engineering techniques is an open research topic. This paper presents a questionnaire for evaluating and comparing development methods and tools.

Keywords: Multiagent systems, software engineering, development tools.

1 Introduction

Due to the great number of methods and frameworks to develop multiagent systems (MAS), the selection of one or another multiagent development tool is a very hard task. In the last few years the evaluation of MAS software engineering techniques has gained the research community attention. Some works [5,7] focus their efforts on the analysis of methodologies, but do not analyze the tools that provide support for these methodologies. Other works like [1,8] analyzes environments for developing software agents, but do not take into account the gap between the methodology and modeling tool. Works like [2,9] only analyze methodologies but they do not only provide a list of concepts to analyze. They facilitate the evaluation task providing a questionnaire which usage makes the answers more concrete and simplifies the evaluation process.

In our work we try to contribute with a framework to evaluate MAS software engineering development methods and tools that deals with those open issues in the field of software engineering MAS evaluation. The initial results of the definition of the evaluation criteria are presented in [3] and validated analyzing the Ingenias methodology and development kit in [4]. The main goal of this paper is to complete our previous work offering an improved and updated selection of the evaluation criteria and defining a complete evaluation questionnaire that will help in facilitating, standardizing and simplifying the evaluation task.

2 Definition of the Evaluation Criteria

From a software engineering perspective, solving a problem should encompass the steps from problem realization, requirements analysis, architecture design and implementation [5]. In order to cover all the necessary method characteristics and tool features for all the stages of the development process, the evaluation criteria are structured in two main parts (See Figure 1): (1) Methodology and Modeling language; (2) Development tool that involves the Modeling tool and the Implementation tool. Furthermore, these criteria focus their attention in the gap between the theoretical guidelines of the methodologies and what can be modeled in the MAS development environment; and the gap between the model and the final implementation. The rest of the section details the evaluation criteria following this classification. In order to get a complete and easy to use evaluation framework the evaluation criteria is presented as a questionnaire. Due to the space limitations, only a brief introduction to each category and some remarks about some criteria is presented.

2.1 Methodology and Modeling Language

This section defines a process for evaluating the methodologies and the modeling languages, comparing strengths, weaknesses and identifying ways to improve on a particular methodological feature. The methodology evaluation criteria is divided into five categories (Figure 1): Concepts and properties, Model related criteria, Process related criteria, Pragmatic features criteria, Supportive feature criteria. This classification is based on [9,5,10].

Concepts and properties: This section analyzes whether or not a methodology adheres to the features of agents and MAS (Table 1). Despite the confusion in the definition of what is an agent and a MAS, these issues are commonly accepted and used [6].

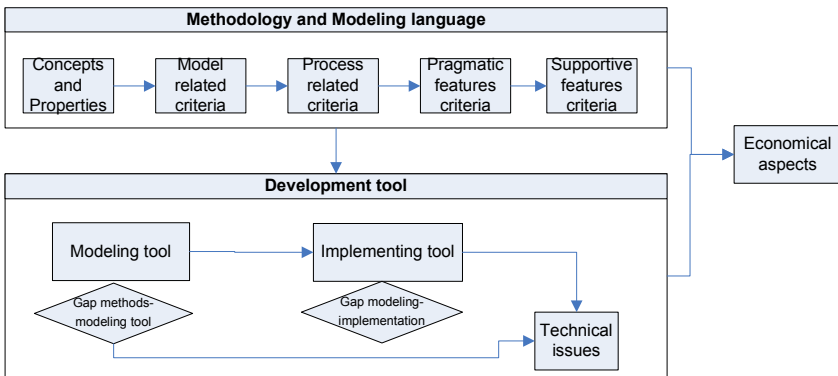


Fig. 1. Evaluation criteria classification

Table 1. Concepts and proprieties questionnaire

Agent architecture: Is the methodology or the modeling language focused on an specific agent architecture?
Platform dependency: Is the development phase of the methodology focused on a specific deployment platform?
Autonomy: Can the models support and represent the autonomous feature of agents? Which technology is used to represent it?
Reactivity: Can the models support and represent the agent's ability to respond in a timely manner to changes in the environment?
Proactiveness: Can the models support and represent the agent's ability to pursue goals over time?
Cooperative behaviour: Can the models support and represent the ability to work together with other agents to achieve a common goal?
Communication ability: Can the models support and represent the ability to communicate with other agents?
Communication language: The communication language used by the agents is based on: O Signals (i.e low level languages) O Speech acts O Other, please specify ____
Mental attitudes: Can the models support and represent the use of agent mental attitudes like beliefs, desires and intentions?
Adaptability: Can the models support and represent the ability of the agents to learn and improve with experience?
Temporal continuity: Can the models support and represent temporal continuity of agents?
Deliberative capability: Can the models support and represent the capability of the agent to select some possible plans to solve a problem and deliberate to choose the most appropriate in each situation?

Model related criteria: The criteria presented in Table 2 deal with various aspects of a methodology's models and notational components, including the concepts represented by the models, and their expressiveness and other software engineering issues.

Table 2. Model related criteria questionnaire

Modeling language representation: Which kind of representation is used? O Formal O Informal O Mixed
Metamodels: Is the methodology based on metamodels?
Kind of models: Which models are used? Please add a brief description of each one.
Models dependence: Is there a high dependence between the models?
Complete notation: Can the modeling language support and represent all the concepts expressed by the methodology?
Concurrency: Can the models support and represent concurrent processes and the synchronization of concurrent processes?
Clarity: Is the number of concepts expressed in a single diagram manageable?
Completeness: Can the models support and represent all necessary concepts that describe the associated methodologies?
Protocols: Can the models support and represent protocols, i.e., the definition of the allowable conversations in terms of the valid sequences of messages?
Different levels of abstraction: Does the methodology and its models provide support for model at various levels of abstraction and detail?
Human Computer Interaction: Does the methodology and its models provide support for model user interface and system-user interaction?
Modularity: Does the methodology and its models provide support for modularity of design components?
Extensible: Is the modeling language extensible?
Environment: Does the modeling language provide support for model the environment of the agents and the agent effectors and perceptors?
Dynamic environment: Does the modeling language provide support for model environment changes?
Resources: Does the modeling language provide support for model agent external and internal resources, and their restrictions?

Process related criteria: The questions presented in Table 3 analyze how the methodology covers the different stages of the development process and which guidelines offers.

Table 3. Process related criteria questionnaire

Development lifecycle: Which software-development process follows the methodology? O Iterative O Waterfall O Others ____
Coverage of the lifecycle: What phases of the lifecycle are covered by the methodology? Which is the level of support of each phase?
Development approach: Which development approach is supported? O Top-down approach O Bottom-up approach O Both O Indeterminate
Approach towards MAS development: Is the methodology OO-based or knowledge-engineering based? O OO-based O Knowledge-engineering based O Other ____
Application domain: Is the methodology applicable to a specific application domain?
Model-central element: Which is the model-central element? O Agents O Organizations O Services O Other ____
Guidelines: Does the methodology and the modeling language offer guidelines for the following issues? - <i>Consistency guidelines:</i> Are there guidelines and techniques for consistency checking both within and between models? - <i>Consistency between levels of abstractions:</i> Are there guidelines to ensure consistency between levels of abstractions within each model and between different models? - <i>Estimating guidelines:</i> Are there guidelines to estimate the number of agents required, the implementation time or other useful features? - <i>Support for decisions:</i> Are there guidelines about when to move between phases? - <i>Model derivation:</i> Are there guidelines to transform models into other models, or partially create a model from information present in another? - <i>Support for verification and validation:</i> Does the methodology contain rules to allow for the verification and validation of correctness of developed models and specifications?
Client communication: Does the methodology provide support and facilitate communication between designers and users?
Models Reuse: Does the methodology provide, or make it possible to use, a library of reusable models?

Pragmatic related criteria: The criteria presented in Table 4 assess software engineering features that evaluate the techniques provided by the methodology for the execution of its process steps and for the development of its models.

Table 4. Pragmatic related criteria questionnaire

Unambiguity: Has the notation of the modeling language unambiguous mapping of concepts to symbols, uniform mapping rules and no overloading of notation elements?
Preciseness of models: Are the mapping between notation and semantics clearly defined?
Expressiveness: Are the models capable of capturing each concept at a great level of detail?
Consistency checking: Does the model representation allow for consistency checking between their elements?
Notation simplicity: Is notation semantically and syntactically simple across models?
Refinement: Does the methodology and its models provide support for a refinement-based design approach?
Documentation: Are the use and features of the tool well documented?
Examples Is any complete example of the use of the tool available? MASDK.

Supportive related criteria: The criteria presented in Table 5 include high-level and complementary features of MAS and the offered support to the integration with other techniques and technologies.

Table 5. Supportive related criteria questionnaire

Open systems: Does the methodology and the modeling language provide support for open systems with heterogeneous agents?
Security: Does the methodology and the modeling language provide support for security techniques in agent applications?
Scalability: What size of MAS is the methodology suited for? O Small O Medium O Large O All
Support for mobile agents: Do the methodology and the modeling language support the use of mobile agents in MAS?
Support for ontology: Do the methodology and the modeling language support the use/integration of ontology in MAS?
Support for MAS organizations: Do the methodology and the modeling language support the use of organizational MAS?
Support for the integration with web services: Do the methodology and the modeling language support the integration with web services?

2.2 Development Tools

This section analyzes how the specifications of the methodology are supported by the modeling tool and which implementation facilities offer the implementation tool. Finally, the gap between the methods and the models and the gap between the models and the final implementation are reviewed. As shown in Figure 1, the analysis of the development tool is divided in five categories: (1) Modeling tool; (2) Gap between methods and the modeling tool; (3) Implementing tool; (4) The gap between modeling and implementation; and (5) Technical issues.

Modeling tool: The modeling tool should allow the transformation of the abstract concepts and ideas of the methodology into diagrams and models using a specific modeling language (Table 6).

Table 6. Modeling tool questionnaire

Kind of representation supported: Which kind of representation is used? O Graphics O Formal languages O Mixed
Automated generation of parts of the models: Does the modeling tool automatically generate parts of the models?
Automated generation of models from requirements: Does the modeling tool able to read a definition of the requirements and generate part of the models?
Store model language: Does the modeling tool store the models in a standard language?
Support for model checking: Does the modeling tool include support and tools to apply model checking?
Support for ontology: How the modeling tool supports the definition of the ontology? O Do not support O Allow implementing it O Allow importing it from other programs Which are:
Static verification tools: Does the tool provide an static verification? - <i>Detect inconsistencies:</i> Does the modeling tool detect inconsistencies within and between models? - <i>Detect incompleteness:</i> Does the modeling tool detect incompleteness within and between models? - <i>Propose solutions:</i> Does the modeling tool proposes solutions when detects an error within or between models? - <i>Others:</i> Does the modeling tool offer other static verification tools?
Dynamic verification tools: Does the tool test the behaviour of the applications using simulations, i.e. simplified system prototypes?

Gap between methods and the modeling tool: This category analyzes how the modeling tool covers the specific features of a methodology (Table 7), the

gap between what is defined by the methodology and what can be modeled using the modeling tool should be as little as possible.

Table 7. Gap between methods and the modeling tool questionnaire

Support for the modeling language: Does the modeling tool support all the features and concepts that define the modeling language?
Support for the kind of representation: Does the modeling tool support all the types of representation used by the modeling language?
Lifecycle coverage: Does the modeling tool support all the development stages supported by the methodology?
Development guidelines: Does the modeling tool integrate the methodology development guidelines? - <i>Consistency guidelines:</i> Are the guidelines and techniques for consistency checking both within and between models integrated in the modeling tool? - <i>Consistency between levels of abstractions:</i> Are the guidelines to ensure consistency between levels of abstractions within each model and between different models integrated in the modeling tool? - <i>Estimating guidelines:</i> Are the guidelines to estimate the number of agents required, the implementation time or other useful features integrated in the modeling tool? - <i>Support for decisions:</i> Are the guidelines about when to move between phases integrated in the modeling tool? - <i>Model derivation:</i> Are the guidelines to transform models into other models, or partially create a model from information present in another, integrated in the modeling tool? - <i>Support for verification and validation:</i> Are the rules that allow the verification and validation of correctness of developed models and specifications integrated in the modeling tool?

Implementing tool and programming language: This category analyzes which support offers the implementing tool to develop MAS, and also, it analyzes traditional software engineering features of this kind of tools (Table 8).

Table 8. Implementing tool questionnaire

Platform dependent: Does the tool allow implementing code for any MAS-execution platform? O Yes O No, only for: _____
Debugging facilities Does the tool offer debugging facilities?
Generation of graphical interfaces: Does the tool offer the possibility to generate graphical interfaces?
Limited systems: Does the tool support the development of system with some limitations, i.e., the development of system that are going to be executed in limited devices like mobile phones?
Real time control: Does the tool offer facilities to use real time control techniques?
Security issues: Does the tool offer facilities to include security issues in the code of the MAS?
Physical environment models: Does the tool offer a library of simulators of physical parts of some kinds of systems for testing the functionality and correctness of the developed system?
Utility agents: There are different agents offering services that do not depend on the particular application domain. Does the implementing tool offer the code of some utility agents?
Reengineering: Does the tool offer the possibility to use reengineering techniques?

The gap between modeling and implementation: As is explained in [8,11], it is very important to analyze the gap between what is modeled and what can be finally implemented (Table 9). Most times, the implementation is developed completely manually from the design. This creates the possibility for the design and implementation to diverge, which tends to make the design less useful for further work in maintenance and comprehension of the system [6].

Table 9. Gap between modeling and implementation questionnaire

Match MAS abstractions with implementation elements: Have all the concepts modeled a direct translation into an implementation element?
Automatic code generation: Can the implementing tool read some models and generate parts of the code automatically? Which automatic code generation technology is used? ----
Specific platform facilities: Does the implementing tool provide facilities for developing the code for a specific platform?

Technical issues: The issues presented in Table 10 are traditional software engineering features that are related with the requirements of a tool to be installed, executed and used.

Table 10. Technical issues questionnaire

Programming language: With which programming languages has been implemented the tool?
Installation requirements: - In which platforms can be executed? - Over which operating systems can be executed? - Is it light-weight?
Required expertise: Is it necessary be a expert modeler and developer to use the tool?
Facility to learn: Is the use of the tool easy to learn?
Facility to use: Is the tool easy to use?
Extensibility: Is easy to add new functionalities to the tool?
Scalability: Is the tool ready to develop any scale of applications (small systems or large-scale applications)?
Collaborative development: Does the tool offer support for collaborative development?
Documentation: Are the use and features of the tool well documented?
Examples: Is any complete example of the use of the tool available? MASDK.

2.3 Economical Aspects

These features has to be evaluated both for the methodologies and for the development tools. The criteria presented in Table 11 do not only include the cost of the application, also the vendor organization and the documentation offered is analyzed.

Table 11. Economical aspects questionnaire

License: Under which license is available the evaluated method or tool?
Cost of the application: What is the cost a license of the application?
Cost of it documentation: What is the cost of it documentation?
Vendor organization: Which is the vendor organization responsible of the evaluated method or tool? O Industrial vendor: ___ O Academical vendor: ___
Updates: Is the evaluated method or tool a static and definitive version or is an open project with regular updates?
Technical service: Does the vendor organization provide technical service?
Examples of academical use: How many academical applications have been developed using the evaluated method or tool? O Any O One O 2-5 O 5-10 O More than 10
Examples of industrial use: How many industrial applications have been developed using the evaluated method or tool? O Any O One O 2-5 O 5-10 O More than 10

3 Conclusions

This paper presents a complete evaluation questionnaire for MAS software engineering. The completeness and relevance of the presented evaluation criteria are reflected via its attention to both system engineering dimensions and MAS features, and because it analyzes the MAS development process from the requirement stage to the implementation of the final code taking into account the most important features and tools involved in the process. Furthermore, the gap between methods and modeling issues, and the gap between the models and the final implementation is studied. The evaluation criteria is presented as a questionnaire in order to facilitate the evaluation task and to make the evaluation process more concrete and comparable.

With the results of the evaluations of such a framework a developer could select the more appropriate MAS software engineering methods and tools for the particular system to develop. Furthermore, this questionnaire summarizes the most important issues for developing MAS, so it could be used for MAS software engineering developers to detect and improve lacks in their methods and tools.

Acknowledgements

This work is partially supported by the TIN2006-14630-C03-01, PAID-06-07/3191 projects and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

References

1. Bitting, E., Carter, J., Ghorbani, A.A.: Multiagent System Development Kits: An Evaluation. In: Proc. CNSR 2003, May 15-16, pp. 80–92 (2003)
2. Dam, K.H.: Evaluating and Comparing Agent-Oriented Software Engineering Methodologies. Master's thesis, RMIT University, Australia (2003)
3. Garcia, E., Giret, A., Botti, V.: Evaluating mas engineering tools. In: International Conference on Evaluation of Novel Approaches to Software Engineering, pp. 181–1874 (2008)
4. Garcia, E., Giret, A., Botti, V.: On the evaluation of mas development tools. In: International Conference on Artificial Intelligence in Theory and Practice (IFIP AI 2008). Springer, Boston (in press, 2008)
5. Lin, C., Kavi, K.M., Sheldon, F.T., Daley, K.M., Abercrombie, R.K.: A methodology to evaluate agent oriented software engineering techniques. In: HICSS 2007, p. 60. IEEE Computer Society, Los Alamitos (2007)
6. Rafael, M.D., Bordini, H., Winikoff, M.: Current issues in multi-agent systems development. In: Post-proceedings of the Seventh Annual International Workshop on Engineering Societies in the Agents World, pp. 38–61 (2007)
7. Sturm, A., Shehory, O.: A framework for evaluating agent-oriented methodologies. In: Giorgini, P., Henderson-Sellers, B., Winikoff, M. (eds.) AOIS 2003. LNCS, vol. 3030, pp. 94–109. Springer, Heidelberg (2004)
8. Sudeikat, J., Braubach, L., Pokahr, A., Lamersdorf, W.: Evaluation of agent-oriented software methodologies examination of the gap between modeling and platform. In: Odell, J.J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 126–141. Springer, Heidelberg (2005)

9. Tran, Q., Low, G.: Comparison of ten agent-oriented methodologies. In: Henderson-Sellers, B., Giorgini, P. (eds.) *Agent-Oriented Methodologies*, pp. 341–367. Idea Group Publishing (2005)
10. Wooldridge, M., Ciancarini, P.: *Agent-Oriented Software Engineering: The State of the Art*. In: Ciancarini, P., Wooldridge, M.J. (eds.) *AOSE 2000*. LNCS, vol. 1957, pp. 1–28. Springer, Heidelberg (2001)
11. Xue, X., Zeng, J., Liding, L.: Towards an engineering change in agent oriented software engineering. In: *ICICIC 2006: Proceedings of the First International Conference on Innovative Computing, Information and Control*, pp. 225–228. IEEE Computer Society, Los Alamitos (2006)

From Obligations to Organizational Structures in Multi-Agent Systems

J. Octavio Gutiérrez-García^{1,2}, Jean-Luc Koning¹, and Félix F. Ramos-Corchado²

¹ Grenoble Institute of Technology, LCIS Research Laboratory,

50, rue Barthélémy de Laffemas, BP 54, 26902 Valence Cedex 9, France

² Centro de Investigación y de Estudios Avanzados del IPN, Unidad Guadalajara,

Av. Científica 1145, Col. El Bajío, Zapopan 45010, Jalisco, México

jgutierrez@gdl.cinvestav.mx, jean-luc.koning@grenoble-inp.fr,

framos@gdl.cinvestav.mx

Abstract. The achievement of common objectives in multi-agent systems is only possible through interaction and coordination; in order to implement both aspects in a effective manner, rules to direct the behavior of a group of agents are necessary, however, existing rules are usually static, inflexible, and inappropriate for large systems, where dynamic interaction takes place. We propose modeling agent behavior by means of obligations, utilized as social norms, delineating agents' roles as independent components, which can be grouped into organizational structures. Moreover, such organizations can be deployed on a service oriented platform, where the composition of organizations leads to the creation of new services.

Keywords: Multi-agent systems, interaction protocols, and social norms.

1 Introduction

Agent interaction is generally governed by a set of rules, which are called interaction protocols (IPs). The role played by IPs is vital to lead agent coordination coherently. However, the rigid structure provided by current modeling techniques (e.g. finite state automata) restrain the autonomy of the agents, limiting their proactivity. Therefore more flexible and dynamic rules are required.

In recent times, a different trend to define rules, based on social norms, has emerged. In [4] and [8] norms are expressed as social commitments, which represent a promise of an agent to reach certain state of affairs to another agent. In [1] norms are defined as permissions and obligations. In [2] and [6] norms are grouped in contracts. Contracts are enhanced with agent societies, which are composed of social organizations [3]. An organization is an association of agents, with an objective, that is regulated by control mechanisms.

In here, we propose an interaction framework, inspired by social mechanisms of control. As main element of control, we use social obligations. Obligations are used to constrain and direct the functions of roles. Afterwards, interrelated roles are grouped into organizations, which are deployed on a service oriented platform, where the composition of non-existing services can take place.

This paper is structured as follows: In section 2, obligations are formalized; section 3 presents the definition of roles; section 4 explains the organizational structure; section 5 illustrates the use of our modeling technique; section 6 describes how organizations are composed; in section 7, a comparison with similar approaches is presented; finally in section 8, we give some concluding remarks and the future work.

2 Definition and Management of Obligations

We define obligations as moral impositions to oneself to reach a state of affairs, demanded by a social force. An agent adopts obligations with the aim of exchanging them for rights. The symbolic representation of an obligation is given by $O(agt, f)$, which means that an agent agt is obliged to reach the state of affairs f . Obligations can express prohibitions as obligations to not reach a state of affairs, represented by $O(agt, \neg f)$. The states of an obligation are: allocated and released.

A special kind of obligation is the conditional obligation (CO), meaning that an agent will be obliged to reach a state of affairs, whenever certain condition holds, this is represented by $CO(agt, f, g)$, where g represents the necessary condition. A CO is only active if the state of affairs g is held while the obligation is allocated. Hence the states of a CO are: allocated-active, allocated-inactive, and released.

To formalize obligations and their management, we have selected the Event Calculus (EC) formalism, for its simple and intuitive definition of actions.

2.1 The Event Calculus Formalism

The EC is a temporal formalism that provides a structure to reason about events and the time when these occur [7]. An EC predicate can contain three kinds of elements: an action $a()$, a fluent f that represents a property that is affected by the action at certain point of time t . We use the simple EC presented in [9]. The EC predicates utilized are:

- $Initiates(a(), f, t)$ means that the fluent f holds after the execution of $a()$ at time t .
- $Terminates(a(), f, t)$ denotes that the fluent f does not hold after the execution of the action $a()$ at time t .
- $HoldsAt(f, t)$ means that the fluent f holds at time t .
- $Happens(a(), t)$ means that the action $a()$ is executed at time t .
- $InitiallyP(f)$ and $InitiallyN(f)$ means that f holds or does not hold, from $t=0$.

2.2 Operations over Obligations

Next, we will proceed with the formalization of the operations that can be applied over obligations and/or conditional obligations, indistinctively. The sentence structure is based on the work presented in [10], while semantics were overlapped by the one provided by obligations.

Creating obligations. The creation of an obligation to reach a state of affairs \mathbf{f} , by performing an action $a()$, is shown next:

$$\text{CreateO}(a(\text{agt}), O(\text{agt}, f), t): \{\text{Happens}(a(\text{agt}), t) \wedge \text{Initiates}(a(\text{agt}), O(\text{agt}, f), t)\} . \quad (1)$$

Where the predicate $a(\text{agt})$ denotes that the action $a()$ is performed by the agent agt ; the fluent $O(\text{agt}, f)$ represents that the agent agt is obliged to reach the state of affairs f , and t represents the instant when the action is performed.

An obligation can also be induced to another agent by means of a social force. This is defined as follows:

$$\text{CreateO}(a(\text{agt}_1), O(\text{agt}_2, f), t): \{\text{Happens}(a(\text{agt}_1), t) \wedge \text{Initiates}(a(\text{agt}_1), O(\text{agt}_2, f), t) \wedge \text{HoldsAt}(\text{AllowedBy}(\text{agt}_2, O(\text{agt}_2, f)), t)\} . \quad (2)$$

If the predicate $\text{AllowedBy}(\text{agt}_2, O(\text{agt}_2, f))$ is evaluated true, implies that the agent agt_2 accepts the obligation $O(\text{agt}_2, f)$ induced by agt_1 .

Releasing obligations. The releasing of an obligation takes place when the state of affairs f has been reached by the agent agt . This is stated as follows:

$$\text{ReleaseO}(a(\text{agt}), O(\text{agt}, f), t): \{\text{Happens}(a(\text{agt}), t) \wedge \text{Initiates}(a(\text{agt}), f, t) \wedge \text{Terminates}(a(\text{agt}), O(\text{agt}, f), t)\} . \quad (3)$$

Obligations can also be released by another agent, in such a case:

$$\text{ReleaseO}(a(\text{agt}_1), O(\text{agt}_2, f), t): \{\text{Happens}(a(\text{agt}_1), t) \wedge \text{Initiates}(a(\text{agt}_1), f, t) \wedge \text{Terminates}(a(\text{agt}_1), O(\text{agt}_2, f), t)\} . \quad (4)$$

Canceling obligations. When an obligation is canceled, new obligations can emerge to compensate the cancelation; moreover, canceling an obligation can produce a cascade of cancelations. This is represented in the next definition:

$$\text{CancelO}(a(\text{agt}), O(\text{agt}, f), F^c, F^d, t): \{\text{Happens}(a(\text{agt}), t) \wedge \text{Terminates}(a(\text{agt}), O(\text{agt}, f), t) \wedge \text{Initiates}(a(\text{agt}), O(\text{agt}, c), t) \mid c \in F^c \wedge \text{Terminates}(a(\text{agt}), O(\text{agt}, d), t) \mid d \in F^d\} . \quad (5)$$

Where F^c and F^d denote the set of additional obligations to compensate and to cancel, respectively. Both F^c and F^d can be empty sets.

To ensure an appropriate management of the creation, releasing and cancelation of obligations, the following rules are required:

1. $\text{CreateO}(a(\text{agt}), O(\text{agt}, f), t) \leftarrow \neg \text{HoldsAt}(O(\text{agt}, f), t) \wedge \text{Happens}(a(\text{agt}), t)$
2. $\text{ReleaseO}(a(\text{agt}), O(\text{agt}, f), t) \leftarrow \text{HoldsAt}(O(\text{agt}, f), t) \wedge \text{Happens}(a(\text{agt}), t)$ (6)
3. $\text{CancelO}(a(\text{agt}), O(\text{agt}, f), F^c, F^d, t) \leftarrow \text{HoldsAt}(O(\text{agt}, f), t) \wedge \text{Happens}(a(\text{agt}), t)$.

The first rule indicates that an obligation can be created only if it is not already held; the second and third rules, state that an obligation can be released or canceled, respectively, only if the obligation is held.

3 Role Definition

With obligations, we are able to model the roles that each agent can play inside an organization. A role has 7 elements: Interaction context, domain knowledge, initial obligations, actions, effects of cancelling obligations, interaction state and a priority relation. Next, a detailed definition of each element:

a) Interaction context (W). It contains the obligations that agents can commit to, and those ones that can induce to other roles. An obligation can have 4 properties:

- *Self-induced* (\boxtimes). The obligation can be acquired by agent's own decision.
- *Out-in induction* (\downarrow). The obligation can be externally induced by another agent.
- *In-out induction* (\uparrow). The agent has the power to induce the obligation to others.
- *Final* (*). The obligation can be released by the agent, without intermediaries.

The interaction context can be used as an interface to couple with other roles.

b) Domain knowledge (K). Obligations require of domain knowledge variables, to specify the properties and/or level of the obligations that an agent is committed to.

c) Initial obligations (I). When an agent instantiates a role, a set of initial obligations is assigned to it, by the simple fact of playing the role.

d) Actions (A). An action definition has three elements: preconditions, the action itself, and its effects. Next, a detailed definition of each element is presented:

- *Preconditions:* The preconditions are defined in terms of obligations, and are represented by a set of conjunctions and disjunctions of elements of four different classes, for a certain state of affairs f .

1. When a state of affairs has to be already accomplished: $\text{HoldsAt}(f, t_1)$.

2. When is enough that someone is committed to reach f : $\text{HoldsAt}(\text{O}(\text{agt}, f), t_1)$.

3. When an agent can decide between accepting the accomplishment of a state of affairs or considering enough that another agent is obliged to reach it: $(\text{HoldsAt}(f, t_1) \oplus \text{HoldsAt}(\text{O}(\text{agt}, f), t_1))$.

4. When the action has to be executed in a predefined period of time, this is expressed in the following way $(t_1 [< | > | =] t_2)$.

- *Actions.* Several messages or actions sent/performed by an agent can be grouped for the same set of preconditions, but each action should have its own effects.

- *Effects.* Releasing and creation of obligations.

e) Effects of canceling obligations (C). For each possible obligation that can be created during the interaction, effects for its cancelation have to be defined:

$$C = \{ \text{CancelO}(a(\text{agt}), \text{O}(\text{agt}, f), F^c, F^d, t) \mid \forall \text{O}(\text{agt}, f) \text{ that can be instantiated} \} . \quad (7)$$

f) Interaction state (S). The current interaction state of an agent is determined by the obligations that have to accomplish at certain moment. This is denoted as follows:

$$S = \{ \text{HoldsAt}(\text{O}(\text{agt}, f), t) \mid \forall f \text{ that the agent agt is obliged to reach} \} . \quad (8)$$

g) Priority relation (P). Since not all obligations are equally important, a preference relation should be provided; such relation has two operators:

- $O_1 \gg O_2$ denotes that the releasing of O_1 is preferred over the releasing of O_2
- $O_1 \parallel O_2$ denotes that there is not preference between complying with O_1 or O_2 .

4 Organizations

Interrelated roles are grouped into organizations with the aim of providing a service, adding a new level of encapsulation to the interaction. We define an organization *Org* as a 6-tuple (W, K, G, P, R, S) , where *W* denotes Interaction Context, *K* Domain knowledge, *G* Group of agents, *P* Proprietor of the organization, *R* Set of roles, and *S* State of the organization. A detailed definition of each element is presented next:

a) *Interaction context (W)*. The interaction context is defined by the union of the interaction contexts of all the roles that belong to the organization, providing a description of its function, as well as an interface to couple with other organizations.

b) *Domain knowledge (K)*. The domain knowledge of the organization is defined by the union of the domain knowledge of each role that belongs to it.

c) *Group of agents (G)*. An agent *agt* is a member of an organization in a determined time *t*, if it complies with the next: $\exists f \text{ Holds}(O(\text{agt}, f), t)$.

d) *Proprietor of the organization (P)*. The proprietor of an organization is the main sustentation of the organization; it should provide the role's definitions to new members, publish the interaction context, and keep track of the interaction state.

e) *Set of roles (R)*. An organization contains a set of roles, which provides an abstraction of all the functionalities of the organization.

f) *State of the organization (S)*. The current state of an organization is determined by the obligations that each member of the organization has:

$$\text{State} = \{ \forall \text{agt}, \forall f, \text{HoldsAt}(O(\text{agt}, f), t), \text{ for a given time } t \} . \quad (9)$$

BUYER (B)

$W_B = \{ \text{Pay}(x) \supset^*, \text{SendInfo}(x) \uparrow, \text{SendDetails}(x) \uparrow, \text{Deliver}(x) \uparrow, \text{SendReceipt}(x) \uparrow, \text{PayRetractionFee}(x) \supset^*, \text{ReturnProduct}(x) \supset^* \}$

$K_B = \{ \text{list-products}, \text{details}, \text{mod}, \text{qty}, \text{price}, \text{deadlineP} \}$

$I_B = \{ \emptyset \}$

$P_{B,1} = \{ \emptyset \}$

$A_{B,1} = \text{Request}(\text{list-products})$

$E_{B,1} = \{ \text{CreateO}(\text{Request}(\text{list-products}), O(S, \text{SendInfo}(x)), t) \}$

$P_{B,2} = \{ \emptyset \}$

$A_{B,2} = \text{Request}(\text{details}, \text{mod})$

$E_{B,2} = \{ \text{CreateO}(\text{Request}(\text{details}, \text{mod}), O(S, \text{SendDetails}(x)), t) \}$

$P_{B,3} = \{ \emptyset \}$

$A_{B,3} = \text{Accept}(\text{qty}, \text{mod}, \text{price}, \text{deadlineP})$

$E_{B,3} = \{ \text{CreateO}(\text{Accept}(\text{qty}, \text{mod}, \text{price}, \text{deadlineP}), O(B, \text{Pay}(x)), t), \text{CreateO}(\text{Accept}(\text{qty}, \text{mod}, \text{price}, \text{deadlineP}), O(S, \text{Deliver}(x)), t) \}$

$P_{B,4} = \{ \text{HoldsAt}(O(B, \text{Pay}(x)), t) \wedge (\text{HoldsAt}(O(S, \text{Deliver}(x)), t) \oplus \text{HoldsAt}(\text{Deliver}(x), t)) \wedge (t < \text{deadlineP}) \}$

$A_{B,4} = \text{MakePayment}(\text{price})$

$E_{B,4} = \{ \text{ReleaseO}(\text{MakePayment}(\text{price}), \text{Pay}(x), t), \text{CreateO}(\text{MakePayment}(\text{price}), O(S, \text{SendReceipt}(x)), t) \}$

$P_{B,5} = \{ \text{HoldsAt}(O(B, \text{PayRetractionFee}(x)), t) \}$

$A_{B,5} = \text{MakePayment}(\text{price}/10)$

$E_{B,5} = \{ \text{ReleaseO}(\text{MakePayment}(\text{price}/10), \text{PayRetractionFee}(x), t) \}$

$C_B = \{ \text{CancelO}(\text{Withdraw}(\text{qty}, \text{mod}, \text{price}, \text{deadlineP}), O(B, \text{Pay}(x)), \{ O(B, \text{PayRetractionFee}(x)), \text{CO}(B, \text{ReturnProduct}(x), \text{Deliver}(x)) \}, \{ O(S, \text{Deliver}(x)), t \}) \}$

Priority Relation_B = $\text{Pay}(x) \gg \text{ReturnProduct}(x) \gg \text{PayRetractionFee}(x)$

NOTE: P, A, and E Stand for preconditions, actions, and effects, respectively.

SELLER (S)

$W_S = \{ \text{SendInfo}(x) \downarrow^*, \text{SendDetails}(x) \downarrow^*, \text{Deliver}(x) \downarrow^*, \text{SendReceipt}(x) \downarrow^*, \text{Advertise} \supset^*, \text{OfferItems}(x) \supset^*, \text{RefundMoney}(x) \supset^* \}$

$K_S = \{ \text{list-products}, \text{details}, \text{mod}, \text{qty}, \text{price}, \text{deadlineD}, \text{orderId} \}$

$I_S = \{ \text{Advertise} \}$

$P_{S,1} = \{ \text{HoldsAt}(O(S, \text{SendInfo}(x)), t) \}$

$A_{S,1} = \text{Reply}(\text{list-products})$

$E_{S,1} = \{ \text{ReleaseO}(\text{Reply}(\text{list-products}), O(S, \text{SendInfo}(x)), t) \}$

$P_{S,2} = \{ \text{HoldsAt}(O(S, \text{SendDetails}(x)), t) \}$

$A_{S,2} = \text{Reply}(\text{details}, \text{mod})$

$E_{S,2} = \{ \text{ReleaseO}(\text{Reply}(\text{details}, \text{mod}), O(S, \text{SendDetails}(x)), t) \}$

$P_{S,3} = \{ \text{HoldsAt}(O(S, \text{Deliver}(x)), t) \wedge (\text{HoldsAt}(O(B, \text{Pay}(x)), t) \oplus \text{HoldsAt}(\text{Pay}(x), t)) \wedge (t < \text{deadlineD}) \}$

$A_{S,3} = \text{DispatchItems}(\text{qty}, \text{mod}, \text{price})$

$E_{S,3} = \{ \text{ReleaseO}(\text{DispatchItems}(\text{qty}, \text{mod}, \text{price}), O(S, \text{Deliver}(x)), t) \}$

$P_{S,4} = \{ \text{HoldsAt}(O(S, \text{SendReceipt}(x)), t) \}$

$A_{S,4} = \text{EmitReceipt}(\text{orderId})$

$E_{S,4} = \{ \text{ReleaseO}(\text{EmitReceipt}(\text{orderId}), O(S, \text{SendReceipt}(x)), t) \}$

$P_{S,5} = \{ \text{HoldsAt}(O(B, \text{Advertise}), t) \}$

$A_{S,5} = \text{Advertising}()$

$E_{S,5} = \{ \text{ReleaseO}(\text{Advertising}(), O(B, \text{Advertise}), t) \}$

$P_{S,6} = \{ \text{HoldsAt}(O(B, \text{OfferItems}(x)), t) \}$

$A_{S,6} = \text{SuggestProducts}(\text{list-products})$

$E_{S,6} = \{ \text{ReleaseO}(\text{SuggestProducts}(\text{list-products}), O(B, \text{OfferItems}(x)), t) \}$

$P_{S,7} = \{ \text{HoldsAt}(O(S, \text{RefundMoney}(x)), t) \}$

$A_{S,7} = \text{MakePayment}(\text{Price} * \text{Qty})$

$E_{S,7} = \{ \text{ReleaseO}(\text{MakePayment}(\text{Price} * \text{Qty}), O(B, \text{RefundMoney}(x)), t) \}$

$C_S = \{ \text{CancelO}(\text{DeclareOutStock}(\text{mod}), O(S, \text{Deliver}(x)), \{ O(S, \text{OfferItems}(x)), \text{CO}(S, \text{RefundMoney}(x), \text{Pay}(x)) \}, \{ O(B, \text{Pay}(x)), t \}) \}$

Priority Relations_S = $\text{RefundMoney}(x) \gg \text{Deliver}(x) \gg \text{OfferItems}(x) \gg \text{SendReceipt}(x) \gg \text{SendInfo}(x) \parallel \text{SendDetails}(x) \gg \text{Advertise}$

ORGANIZATION STRUCTURE

$W_{\text{MARKET}} = W_B \cup W_S = \{ \text{Pay}(x) \supset^*, \text{SendInfo}(x) \uparrow \downarrow^*, \text{SendDetails}(x) \uparrow \downarrow^*, \text{Deliver}(x) \uparrow \downarrow^*, \text{SendReceipt}(x) \uparrow \downarrow^*, \text{PayRetractionFee}(x) \supset^*, \text{ReturnProduct}(x) \supset^*, \text{Advertise} \supset^*, \text{OfferItems}(x) \supset^*, \text{RefundMoney}(x) \supset^* \}$

$K_{\text{MARKET}} = K_B \cup K_S = \{ \text{list-products}, \text{details}, \text{mod}, \text{qty}, \text{price}, \text{deadlineP}, \text{deadlineD}, \text{orderId} \}$ **Roles** = {Seller and Buyer} **Proprietor** = Seller

Fig. 1. Specification of the market organization

5 Market Organization Example

A market organization is modeled to illustrate our modeling technique. The market organization has two roles: buyer and seller. In this organization, the buyer requests information about certain kind of products, in turn, the seller answers with a list of products; afterwards the buyer asks for a detailed description of a particular model, in which he is interested; after receiving the information, the buyer can decide between buying the product or keep looking for better prices. If the buyer decides to buy the product, he pays for it, and then, the seller delivers the product and sends a receipt. The specification of the market organization is presented in figure 1.

6 Composition of Organizations

Organizations as encapsulated components that provide specific services require appropriate mechanisms in order to be composed to create enhanced services, which will satisfy unpredicted requirements.

In dynamic environments is difficult and expensive to predict every situation, moreover, sometimes is inappropriate to specify every aspect of the interaction, since this could lead to a complex definition of the interaction. To overcome this, we define generic actions that can be dynamically instantiated, whenever an obligation can not be released inside an organization. A generic action is defined as follows:

$$\begin{aligned}
 P: & \{ \text{HoldsAt}(O(\text{Org}_1.\text{Role}, \text{Org}_1.\text{Role}:O_1), t) \} \\
 A: & \text{UnifyOs}(\text{Org}_1.\text{Role}:O_1, \text{Org}_2.\text{Role}:O_2); \quad \text{Org}_2.\text{Role}.\text{Action}(); \\
 E: & \{ \text{ReleaseO}(\text{Org}_2.\text{Role}.\text{Action}(), \text{Org}_1.\text{Role}:O_1, t) \} .
 \end{aligned} \tag{10}$$

Where Org_1 represents the organization where is not possible to release O_1 ; Org_2 represents a variable that will point to another organization where O_1 can be released; the predicate $\text{UnifyOs}(\text{Org}_1.\text{Role}:O_1, \text{Org}_2.\text{Role}:O_2)$ expresses the creation of the linkage, between O_1 and O_2 . Now with both obligations unified, whenever O_2 is released inside Org_2 by the execution of $\text{Org}_2.\text{Role}.\text{Action}()$, the obligation O_1 will also be released inside Org_1 .

In order to locate other organizations, their structures should be deployed on a service oriented platform. Every organization publishes its interaction context in the yellow pages, where agents can search and discover new organizations. We assume that all the agents share the same ontology.

Three steps are required to compose organizations:

1. An agent has to compare the obligation that it wants to release with the interaction contexts published in the yellow pages.
2. Then, with the organization located, an analysis of the role definitions is necessary to decide the role to instantiate. This is performed by looking at the individual interaction context of each role, to see which of them contain an In-Out induction obligation, which matches the Out-In obligation found in step two.
3. The final step requires the identification of the action that releases the obligation in the host organization; this can be done by looking at the effects of the actions.

With the generic action instantiated, the linkage (composition) of both organizations is established, and their functioning isolated from each other.

7 Discussion

An important trend to guide agent interaction by social norms, is the commitment approach. In the work presented in [8] and [10], IPs are specified as commitment machines formalized with EC; an extension of the commitment machines is presented in [5], wherein an algebra for manipulating commitment protocols is proposed. Although the use of commitments provides high flexibility, its general application is still limited, since it is difficult to define a debtor and a creditor for a simple exchange of messages; besides norms like prohibitions and permissions can not be expressed by means of commitments. Therefore, we consider that obligations are more appropriate to express engagement, without requiring a creditor.

With respect to the organizational approach, a work that is close to ours is presented in [1], here executable specifications of organizations are presented. The main difference to our work is the introduction of an institutional power concept; it is said that an agent has the institutional power to perform certain action if it is empowered to do it. Another close work to ours is presented in [3], wherein is proposed a coordination framework based on organizations and commitments; they define hierarchical structures, where high-grade agents can assign or delegate tasks to low-grade agents. In both works [1] and [3], hierarchical structures are established; in this matter, we consider that an organization should be composed of agents with the same level of power, without a central control of coordination, as happens in hierarchical structures; our vision of organizations based on obligations distributes the control of the interaction.

8 Conclusions and Future Work

In this work, we defined obligations as engagements to oneself, demanded by a social force; we introduced their management, using the EC formalism; afterwards an organization structure was defined, which provides an interaction framework instituted by obligations; subsequently a method to compose organizations by means of unifying obligations was presented, enhancing the interaction framework.

The obligations are used as the main element of the interaction framework; their power of expression allows controlling the agent behavior, but respecting at the same time its autonomy. In addition, by modeling IPs by means of obligations, agents' proactivity is encouraged, enabling them to react to new and unforeseen situations.

Organizations, as presented here, are seen as components that can be composed to provide enhanced services; the loosely coupling of organizations makes possible flexible and dynamic interaction, where no previous configuration is required among the parties; as a result, organizations operate as service oriented components that modularize agent interaction, offering scalable properties to MAS.

Our future work will be addressed to develop mechanisms to verify formal properties of the IPs, and mechanisms to check the consistency of the obligations.

References

1. Artikis, A., Sergot, M., Pitt, J.: Specifying Norm-Governed Computational Societies. *ACM Transactions on Computational Logic* (2007)
2. Boella, G., Van der Torre, L.W.N.: A game theoretic approach to contracts in multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* 36(1), 68–79 (2006)
3. Carabelea, C., Boissier, O.: Coordinating Agents in Organizations Using Social Commitments. *Electronic Notes in Theoretical Computer Science* 150(3), 73–91 (2006)
4. Castelfranchi, C.: Commitment: from intentions to groups and organizations. In: *Proceedings of ICMAS 1995*, pp. 41–48. AAAI-MIT Press, Cambridge (2005)
5. Chopra, A.K., Singh, M.P.: Contextualizing commitment protocol. In: *AAMAS 2006*, pp. 1345–1352. ACM Press, New York (2006)
6. Dignum, V., Meyer, J.-J., Weigand, H., Dignum, F.: An organization-oriented model for agent societies. In: *Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications at AAMAS 2002* (2002)
7. Kowalski, R., Sergot, M.: A logic-based calculus of events. *New Generation Computing* 4(1), 67–95 (1986)
8. Mallya, A.U., Singh, M.P.: An algebra for commitment protocols. *Autonomous Agents and Multi-Agent Systems* 14(2), 143–163 (2007)
9. Shanahan, M.P.: The event calculus explained. In: Veloso, M.M., Wooldridge, M.J. (eds.) *Artificial Intelligence Today*. LNCS, vol. 1600, pp. 409–430. Springer, Heidelberg (1999)
10. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: applying event calculus planning using commitments. In: *AAMAS 2002*, pp. 527–534. ACM Press, New York (2002)

Addressing the Brittleness of Agent Interaction

Mohd Fadzil Hassan¹ and Dave Robertson²

¹ Computer and Information Sciences Department,
Universiti Teknologi PETRONAS, Bandar Seri Iskandar
31750 Tronoh, Perak, Malaysia

mfadzil_hassan@petronas.com.my

² Center for Intelligent Systems and their Applications (CISA),
School of Informatics, University of Edinburgh, Scotland, UK
dr@inf.ed.ac.uk

Abstract. The field of multi-agent systems shifts attention from one particular agent to a society of agents; hence the interactions between agents in the society become critical towards the achievement of their goals. We assume that the interactions are managed via a protocol which enables agents to coordinate their actions in order to handle the dependencies that exist between their activities. However, the agents' failures to comply with the constraints imposed by the protocol may cause the agents to have brittle interactions. To address this problem, a constraint relaxation approach derived from the Distributed Partial Constraint Satisfaction Problem (CSP) is proposed. This paper describes the computational aspects of the approach (i.e. specification of a distance metric, searching for a solvable problem and specification of a global distance function).

Keywords: Brittle agent interaction, constraint relaxation for agent interaction, Distributed Partial CSP for computation of agent interaction.

1 Introduction

Depending on the kind of sub-problem interdependencies, the interaction among agents in a multi-agent system (MAS) for a distributed problem solving task can be complex, often requiring a multi-step dialogue. This interaction can be achieved through a protocol that specifies not only the communication of agents, but also the creation and destruction of agents (i.e. agents entering/leaving a MAS), the spatial distribution of agents, as well as synchronization and distribution of actions over time [1]. It generally involves two important elements – the subjects whose activities need to be coordinated (i.e. agents) and the entities between which dependencies arise (i.e. objects of coordination), namely sub-problems handled by the individual agents [2].

The specification of the involved elements and the relationship that exist between them are generally mediated and represented by the notion of *role*. When assuming a role, an agent is in charge of the corresponding task or action, and is entitled to all the authorizations and permissions (and limitations as well) pertaining to its role. The state of the agent interactions is then reflected on the ways the constraints imposed on them are mutually and individually satisfied by the interacting agents.

However, we should realize that when agents interact to solve a particular Multi-agent Agreement Problem (MAP) [3], it is rarely the case that their individual constraints are completely acceptable or completely inconsistent to each other. Rather, it is normally the case that their respective constraints are partially satisfied. Therefore, given that the agents are capable to revise or relax their locally imposed constraints upon encountering an over-constrained situation while participating in the distributed constraint solving process of the MAP, a mechanism is required to accommodate the agents' computational and interactive needs for addressing the problem. In this paper, the focus is on the former.

The remainder of this paper is organized as follows: In section 2 we provide a discussion on the distributed partial CSP and a detailed description of how we use the distributed partial CSP scheme to implement our constraint relaxation approach is given in chapter 3. This paper concludes in section 4.

2 Distributed Partial Constraint Satisfaction Problem

As described in [4], the use of constraint satisfaction techniques in multi-agent systems (MAS) is not new as they have been utilized either as a part of the agents' problem solving apparatus or coordination formalisms as reported in [5, 6]. However, these reported works did not address over-constrained problems. Our approach, on the other hand, considers of integrating distributed partial CSP as part of the constraint handling feature of the MAS interaction protocol system. This is a novel way of providing a more flexible approach for handling constraints during the interactions of heterogeneous and autonomous agents participating in a distributed problem solving task.

A CSP consists of a finite number of variables, each having a finite and discrete set of possible values, and a set of constraints over these variables. A solution to a CSP is an instantiation of all variables for which all the constraints are satisfied. In practice however, it is sometimes the case that certain constraints can be violated occasionally, or weakened to some degree. As conventional CSP techniques lack the mechanisms to accommodate such a notion of constraint handling, this gives rise to the establishment of a niche research area within the constraint satisfaction research field focusing on approaches to solve over-constrained problems, which include distributed partial CSP.

A distributed partial CSP, as abstractly described in [7], consists of:

- A set of agents (problem solvers), $1, 2, \dots, m$
- $\langle (P_i, U_i), (PS_i, \leq), M_i \rangle$ for each agent i
- $(G, (\text{Necs}, \text{Suff}))$, where

For each agent i , P_i is an original CSP (a part of an original distributed CSP), and U_i is a set of *universes*, i.e. a set of potential values for each variable in P_i . Furthermore, (PS_i, \leq) is called a *problem space*, where PS_i is a set of (relaxed) CSPs including P_i , and \leq is a partial order over PS_i . Also, M_i is a locally-defined *distance function* over the problem space. G is a *global distance function* over distributed problem spaces, and $(\text{Necs}, \text{Suff})$ are necessary and sufficient bounds on the global distance between an original distributed CSP (a set of P_i s of all agents) and some solvable distributed CSP (a set of solvable CSPs of all agents, each of which comes from PS_i).

A solution to a distributed partial CSP is a solvable distributed CSP and its solution, where the global distance between an original distributed CSP and the solvable distributed CSP is less than *Necs*. Any solution to a distributed partial CSP will suffice if the global distance between an original distributed CSP and the solvable distributed CSP is not more than *Suff*, and all search can terminate when such a solution is found.

3 Application of Distributed Partial CSP for Addressing Brittleness Problem

In this section, we provide a detailed description of how we use the distributed partial CSP scheme to implement our constraint relaxation approach. This includes a discussion on the following computational aspects of our approach:

- a. The distance metric used (i.e. solution subset distance to compute the degree of constraint relaxation attempted by each individual agent).
- b. Finding a solvable MAP among agents involved in the constraint relaxation process.
- c. The global distance function for agents to compute the best constraint relaxation path to be taken.

3.1 Specification of Distance Metric

This research specializes the solution subset distance metric of [8] for finding a solvable MAP with a minimal degree of constraint relaxation. This is obtained when the agents participating in the constraint relaxation task generate individual problem spaces containing a set of relaxed CSPs, so that the distance between P_2 , a relaxed problem selected from the set, and the original, un-relaxed problem, P_1 , is within a certain bound, according to the specified distance metric. As described in the abstract distributed partial CSP model, the functions to provide distance computation are specified at two separate levels – local and global.

At the local level, we are mainly concerned with the computation of additional solutions introduced due to the individual relaxation attempted by the agents. This is accomplished by comparing P_2 with P_1 each time after a relaxation is performed. Given P_1 , and its corresponding relaxed problem P_2 , the distance metric describes how far the solutions for the two local problems are from each other. This is accomplished by associating the solutions that are already in the original, un-relaxed problem with the one introduced due to relaxation. For instance, the solution subset distance between the two comparable problems P_1 and P_2 is the number of solutions of P_2 which are not solutions of P_1 . Given that the sets S and S' respectively represent the solution sets of the original problem, P_1 , and its corresponding relaxation, P_2 , that is, $S = \text{sols}(P_1)$ and $S' = \text{sols}(P_2)$, where *sols* denotes the solutions to the problem. Then, computation of distance between the two sets (i.e. S and S') not only needs to consider the new additional solutions introduced, but also the existing solutions of the original problem that might be eliminated due to the performed constraint relaxation. Therefore, the equations in figure 1 describe how this is computed, where L is the union of these two components, and the distance, d , is then measured as the cardinality of L .

$$L = (S-S') \cup (S'-S) \quad (1)$$

$$d = |L| \quad (2)$$

Fig. 1. Equations for distance computation

At the global level, we are concerned with the computation of distance of distributed problem spaces. This involves two important steps – first, finding a set of relaxed CSPs allowing for a solvable MAP state to be achieved and second, computing the global distance of the set from their corresponding original problems. Part of the first step also includes the specification of two special bounds to ensure the individual problem space generated by each agent involved in the constraint relaxation interaction is restrained. These two bounds are identified as *necessary* and *sufficient* bounds.

The disruption on agent interactions due to an over-constrained situation will normally result in a partially solvable MAP to be obtained. This MAP contains a set of fully solvable variables, assigned with solution values mutually agreed by all agents. The assignments of these variables are obtained prior to the occurrence of an over-constrained state. This is only possible if there exists a set of variables from the MAP that can be satisfied locally by each agent involved in the problem solving interaction. This set and its assigned solution values are used as the necessary bound. The necessary bound specifies that distributed problem spaces under consideration must all contain solutions that are within the bound. Assuming that all original problems individually specified by the distinct agents at the pre-interaction stage have a set of solutions which has become a fully solvable part of the MAP, then any relaxed problems derived from the originals must contain this set of solutions. This is necessary for preventing any relaxed problem from deviating from an already solvable part of the MAP and effectively restricts the size of the problem space under consideration. This means that any relaxed CSP obtained can only be considered if it satisfies this requirement. In the worst case scenario, this set might be empty, indicating that the interacting agents cannot reach a deal range on any of the variables of the MAP.

A partially solvable MAP also contains a set of non-solvable variables, due to the existence of one or more agents that fail to satisfy their individual constraints concerning these variables during the problem solving interactions. This set of non-solvable variables is specified in the sufficient bound, which describes what needs to be achieved during the constraint relaxation process. A successful value assignment to the set indicates the attainment of a solvable MAP state. A set of relaxed CSPs, obtained from the problem spaces generated by the interacting agents during a particular constraint relaxation cycle, is sufficient if the additional solutions derived from these CSPs allow the initial set of non-solvable variables of the MAP to become solvable.

Both bounds give the required direction to the process of identifying locally relaxed CSPs among the agents from which a consistent, solvable MAP with an acceptable solution subset distance is derived.

3.2 Finding a Solvable MAP

In any MAP solving interaction through a specified protocol, there exist two possible groups of agents. Though in the actual problem solving interaction it might involve

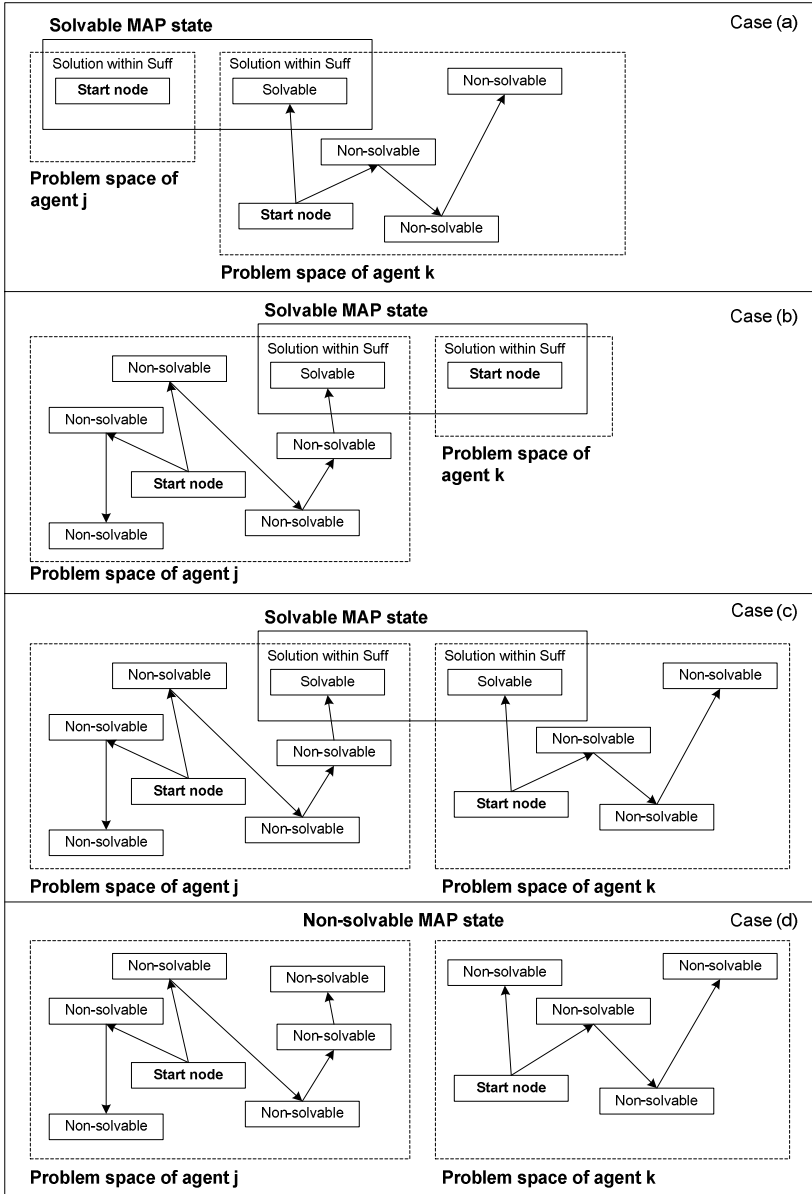


Fig. 2. The search for a solvable MAP state

more than two agents, all the agents can be identified as belonging to either one of these two groups. The first group, J , consists of a set of agents that has completed its part of the protocol in solving and constraining a particular set of variables of the MAP. The second group, K , on the other hand, represents a set of agents whose part in the protocol is incomplete as they cannot satisfy the inter-agent or global constraints imposed on the corresponding set of variables of the MAP. Therefore, the task of finding a solvable MAP given an over-constrained distributed problem solving state, involves a series of local searches on the weakened CSPs provided by these two groups of agents. The weakened CSPs must satisfy the necessary bound, but may not satisfy the sufficient bound. From the view of these two groups of agents who are involved in this collaborative task, the local relaxation process can be thought of as a search which starts from an initial node representing the original CSP of the agent, and follows a path until a solvable MAP state is achieved. The whole searching process is constrained by the specified necessary bound. The process stops when we found a combination of weakened CSPs by the individual agents that satisfy the sufficient bound with some acceptable distance between the derived solution sets. It is then said that a solvable MAP state has been achieved, and there are three possible conditions on how this is accomplished, which are described using agents j and k , instances for agent groups J and K respectively, that is $j \in J$ and $k \in K$:

1. Agent k performs the necessary constraint relaxation on its original CSP, producing a problem space containing the necessary relaxed CSPs, allowing a solvable state to be achieved without the other party, agent j , performing any relaxation on its part as illustrated in figure 2 (a).
2. Agent j performs the necessary constraint relaxation on its original CSP, producing a problem space containing the necessary relaxed CSPs, allowing a solvable state to be achieved without the other party, agent k , performing any relaxation on its part as illustrated in figure 2 (b).
3. Both agents j and k perform the necessary relaxation on their respective original CSPs, where their combined relaxation produces a corresponding set of relaxed CSPs that allow a solvable state to be achieved as illustrated in figure 2 (c).

However, it might also be the case that there exists no improvement towards the achievement of a solvable MAP state after a number of relaxation cycles have been performed. Given this outcome, the relaxation process terminates as it simply indicates that the agents cannot reach an agreement in reconciling their differences as illustrated in figure 2 (d).

3.3 Global Distance Computation

A global distance function, G , is used to measure the global distance between an original distributed CSP (i.e. a set of original CSPs of all agents) and some solvable distributed CSP (i.e. a set of solvable CSPs of all agents, generated by the agents during the constraint relaxation process) in reaching a solvable MAP state. The function can be specified as the following equations:

$$G_{\text{Total}} = \sum_{i=1}^n d_i \quad (3)$$

$$G_{\text{Max}} = \max(d_i) \quad (4)$$

The equation in expression 3 provides the computation for the summation of local distances of all agents participating in the constraint relaxation task, where n is the number of agents involved in the task; d_i is the local distance for each agent i as specified in expression 2 of figure 1, which is the number of additional solutions introduced and existing solutions eliminated due to the relaxation individually performed by each agent on its privately defined finite-domain constraints of the MAP. We search for a combination of relaxed problems generated by the agents that minimize G_{Total} .

The equation in expression 4 provides the computation to find the maximum local distance G_{Max} , given a set of distances, d_i , for each agent i participating in the constraint relaxation task. We search for a combination of relaxed problems generated by the agents with the lowest G_{Max} .

In our work, a two-stage system is employed. In the first stage, we search for a combination of relaxed problems among the agents that produces a minimal G_{Total} . For a search resulting of more than one solution, the system proceeds to the second stage. In the second stage, the G_{Max} for each remaining solution is computed and a solution with the lowest G_{Max} is selected.

4 Conclusion

In this paper, a constraint relaxation approach derived from the distributed partial CSP is proposed for addressing the brittleness of agent interaction in solving an over-constrained MAP. The paper specifically described the computational aspects of the approach (i.e. specification of a distance metric, searching for a solvable problem and specification of a global distance function).

The approach described in the paper has been encoded as an agent interaction protocol using the Lightweight Coordination Calculus (LCC) [9]. In the LCC framework, the protocol language and the expansion engine are written in SICStus Prolog [10] and the message passing system is implemented in Linda [11]. Therefore, we choose to implement our approach in SICStus Prolog to take advantage of the existing code for the LCC basic framework and expansion engine, and ensure smooth interfacing with these components. In addition, a finite-domain constraint solver available in SICStus Prolog (i.e. `clp(FD)`) is used to accommodate the computations on the solution subset distance, necessary and sufficient bounds for the set of problems contained in the agents' problem spaces.

References

1. Bocchi, L., Ciancarini, P.: A perspective on multiagent coordination models. In: Huget, M.-P. (ed.) *Communication in Multiagent Systems*. LNCS, vol. 2650, pp. 146–163. Springer, Heidelberg (2003)
2. Omicini, A., Ossowski, S.: Objective versus subjective coordination in the engineering of agent systems. In: Klusch, M., Bergamaschi, S., Edwards, P., Petta, P. (eds.) *Intelligent Information Agents*. LNCS, vol. 2586, pp. 179–202. Springer, Heidelberg (2003)

3. Modi, P.J., Veloso, M.: Bumping strategies for the multiagent agreement problem. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, The Netherlands (2005)
4. Sycara, K.P.: Multiagent systems. *AI Magazine* 19, 79–92 (1998)
5. Macho-Gonzales, S., Torrens, M., Faltings, B.: A multi-agent recommender system for planning meetings. In: Proceedings of the Workshop on Agent-based Recommender Systems (WARS 2000), Barcelona, Spain (2000)
6. Aldea, A., Lopez, B., Moreno, A., et al.: A multi-agent system for organ transplant coordination. In: Quaglini, S., Barahona, P., Andreassen, S. (eds.) AIME 2001. LNCS, vol. 2101, pp. 413–416. Springer, Heidelberg (2001)
7. Yokoo, M.: Distributed constraint satisfaction: foundations of cooperation in multi-agent systems. Springer, Heidelberg (2001)
8. Bistarelli, S., Freuder, E.C., O’Sullivan, B.: Encoding partial constraint satisfaction in the semiring-based framework for soft constraints. In: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), Boca Raton, FL, USA (2004)
9. Robertson, D.: Multi-agent coordination as distributed logic programming. In: Demoen, B., Lifschitz, V. (eds.) ICLP 2004. LNCS, vol. 3132, pp. 416–430. Springer, Heidelberg (2004)
10. SICS, SICStus Prolog User’s Manual. Stockholm: Swedish Institute of Computer Science (SICS) (1999), <http://www.sics.se/sicstus.html>
11. Carrieno, N., Gelernter, D.: Linda in context. *Communications of the ACM* 32, 444–458 (1989)

Dividing Agents on the Grid for Large Scale Simulation

Dac Phuong Ho, The Duy Bui, and Nguyen Luong Do

College of Technology
Vietnam National University, Hanoi
{phuonghd, duybt}@vnu.edu.vn

Abstract. Multi-agent based simulation is an important methodology that uses models incorporating agents to evaluate research conclusions. When the simulation involves a large number of agent, however, it requires extensively high computational power. In that case, all agents in the simulation model should be distributed in a way so that agents can be run in parallel on multiple computational nodes to gain the required performance speed up. In this paper, we present a framework for large scale multi-agent based simulation on grid. We have modified the desktop grid platform BOINC for multi-agent based simulation. Assuming that the agents interact locally with the environment, we proposed an approach to divide the agents for grid nodes so that we can keep load balancing for the distributed simulation while optimizing the communication between grid nodes and the grid server. We have implemented the food foraging simulation to evaluate the feasibility of the framework.

Keywords: Grid Computing, Multi-agent based simulation.

1 Introduction

The area of multi-agents have received much attention from the research community in the last decade. In a multi-agent system, agents are autonomous computer programs that react to their simulated social and physical environment with regard to their goals. Multi-agents have proven to be useful in many application areas, such as electronic commerce, entertainments, human computer interaction, etc. Besides, multi-agent based simulation is an important methodology that uses models incorporating agents to evaluate research conclusions. When the part of the real world that we want to examine is not accessible, or experimenting with the real system is prohibited due to undesired disturbances, then simulation is a good option. Simulation is a tool for understanding and formalization of a hypothesis that otherwise would have remains very vague. For example, in social science, multi-agent based simulation is a productive and innovative frontier for understanding complex social systems [5], because it allows social scientist to model social phenomena directly in term of social entities and their interaction, in ways that are inaccessible either through statistical or mathematical modeling in close form [3, 4, 8].

Multi-agent based simulation normally refers to time driven event simulation where real world entities are modelled in terms of agents and system behavior emerges from interactions between these agents. When the simulation involves a large number of agent, obviously, it requires extensively high computational power. In that case, all agents of the simulation model should be distributed in a way so that agents can be run in parallel on multiple computational nodes to gain the required performance speed up. Although distributed multi-agent based simulation offers new and interesting opportunities for experimenting with more complex, more realistic, and even more valuable world models, new issues arise with this simulation model. For example, the issues of synchronizing events to ensure causality, monitoring of the distributed simulation state, load balancing, and dynamic resource allocation. So it is difficult for parallel and distributed simulation to achieve the expected performance. In order to exploit the expected computational power of distributed resources, a suitable middleware concept must be in place first. To address the issues of failure resistance, load balancing, transparent executing of large scale simulation experiments in a domain spanning fashion grid technology seems to be most suitable [9, 14].

In the case of multi-agent based simulation installed on a stand-alone computer containing all agents, the agent communication is taking place in the same physical memory. On the Grid, however, the agents are separated in different machines and the communication involves transfer of data through the network. The amount of involved network communication depends very much on how the agents are separated. In this paper, we present a framework for large scale multi-agent based simulation on grid taking into account the dynamic division of agents over the Grid. We have modified the desktop grid platform BOINC [1] for multi-agent based simulation. Assuming that the agents interact locally with the environment, we proposed an approach to divide the agents for grid nodes so that we can keep load balancing for the distributed simulation while optimizing the communication between grid nodes and the grid server. We have implemented the food foraging simulation [12] to evaluate the feasibility of the framework.

The paper is organized as follows. Section 2 gives an overview of related background. We present our platform in Section 3. The experiment with the problem of ant colony system is described in Section 4.

2 Background

2.1 Multi-agent Based Simulation

Multi-agent based simulation is a bottom-up approach to understand the dynamics of a real system via the simulation of many individual agents interacting with each other and with the environment. As agents are heterogeneous, autonomous and pro-active actors, multi-agent based simulation allows modelling of heterogeneous populations in which each agent might have personal motivations and incentives to represent groups and group interactions. This is based on the idea that most work in human organisations is done based on intelligence,

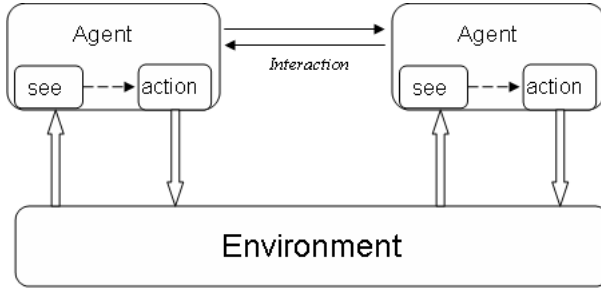


Fig. 1. An overview of a multi-agent system

communication, co-operation, negotiation, and massive parallel processing [7]. With agents having fairly simple behaviour, the complex behaviour of the system can be observed with multi-agent based simulation.

A simple overview of a multi-agent system used for simulation is described in Figure 1. In agent based simulation, there are two types of interaction: interaction between agents and the environment, and interaction among agents. However, in a particular multi-agent based simulation, one often concentrates in one type of interaction and ignore the other. Concentrating on which type of interaction also influences very much the way a multi-agent based simulation system should be distributed.

There are two main disadvantages in using multi-agent based simulation. First, multi-agent based simulation has a higher level of complexity compared to other simulation techniques as all the interactions between agents and between the agent and the environment have to be defined. Second, multi-agent based simulation has high computational requirements.

2.2 Distributed Multi-agent Based Simulation

Multi-agent based simulation should be distributed when there are a large number of agent involves. To present, however, there are not many works focusing on distributed multi-agent based simulation, especially on agent-based simulation on the Grid. Moreover these works only offer pure load balancing facilities which would not allow them to scale well to a large number of agent.

MACE3J [6], a work in progress, uses proxies for interaction of hierarchically ordered simulation nodes. In order to perform progressive scaling of simulation, services in MACE3J are deployed with GT3 using a hierarchical resource federation approach. Arikawa and Murata [2] proposed an implementation of a Grid-based multi-agent simulation by developing applications based on a parameter-sweep approach. Their solution is to examine all given parameter specifications using distributed computing resources on the Grid by developing a computing environment with a task control support framework. In essential, each computer will run only one simulation with particular parameters. The use of the Grid system, however, is just to find out the best set of parameters that is optimal according to defined

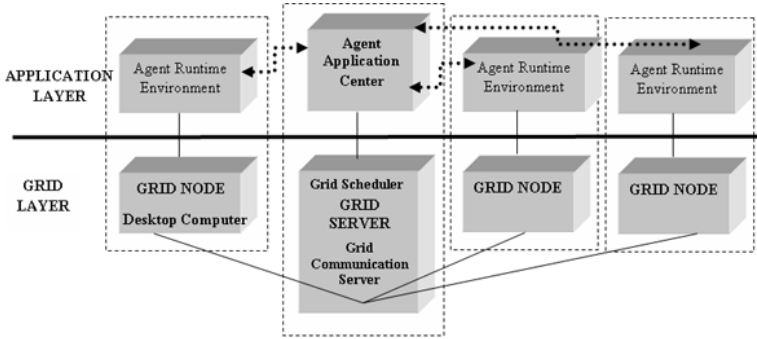


Fig. 2. The desktop grid platform for multi-agent based simulation

criteria while the applications are independent. Timm and Pawlaszczyk [13] proposed a conceptual model for large scale multi-agent simulation to perform the feasibility analysis. However, simulation of realistic scenarios is missing to assess the quality of service as well as performance of grid based simulation. In [10], a model is proposed to predict the performance of multi-agent based simulation on the Grid. With the input parameters of the number of agent, the number of Grid node, the granularity, and the outbound communication rates, the model identified several quantities of interest to be included in performance metrics: simulation run time, thread setup time, and CPU utilization. The experiments show that if the simulation is distributed on M machines, the expected speed up factor will conventionally be equal to M . This factor is possible to archive greater than M with large scale simulations. The measured CPU utilization data shows that due to the communication latency, the system is idle for a considerable part of time in case of small application. The workload of the system increases with large application. The model, however, did not consider the failure which is very common in the Grid system as well as the way to dynamically split agents between computers to ensure the load balancing.

3 A Desktop Grid Platform for Multi-agent Based Simulation

We have modified the desktop grid platform BOINC [1] for multi-agent based simulation with the concentration on load balancing and system performance. The modified platform can be seen in Figure 2. The platform has two layers: the **Grid Layer** and the **Application Layer**.

In this desktop grid platform, any desktop computer in organization can join the grid. Because the owner of the desktop computer can shutdown, reboot, re-configure, connect, disconnect their computer to network at any time, numerous faults can occur with the desktop environment. The **Grid Layer** is there responsible for managing the complexities of the environment in order to provide the **Application layer** above with the simple abstraction. There are several

components in the **Grid Layer**: the **Grid Server**, the **Grid Communication Server**, and the **Grid Scheduler**. The **Grid Communication Server** is responsible for reliable and secure communications between **Grid Nodes** and the **Grid Server**. High quality cryptographic techniques are used to protect application communications and data. The **Grid Scheduler** deals with unique challenges of wide variety of configurations and capabilities of resources, from a cheap PC to expensive clusters in the organization. It accepts units of computation from the Application above, matches them to appropriate client resources, and schedules them for execution. For fault tolerance purposes, after defined amount of time if the **Grid Scheduler** did not get the results from any **Grid Node**, it will resend the task to another **Grid Node** to complete. Each **Grid Node** is installed in a desktop computer. It creates a virtual host for application running. It captures a status of the node (information about physical memory, CPU speed, disk size and free space, software version, data cached, etc.). A **Grid Node** also provides basic facilities for process management including, application initiation and termination, and error reporting. It loads the task from the **Grid Server** and then initializes it. For the Grid Node in BOINC, after executing the task, it sends the result back to the **Grid Server** and terminates the task. This mechanism does not allow the implementation of multi-agent based simulation because the code of the agent should stay in the Grid Node, update the environment status and send back the result frequently. We have modified the Grid Node to allow this.

In the **Application layer**, the **Agent Application Center** is responsible for dividing the application at the agent level. The agents are divided into groups in a location based manner in order to minimize cross influence to the environment between groups, thus minimize the communication between **Grid Nodes** and the **Grid Server** while keeping load balance. A task comprising a group of agents is then sent to the **Grid Scheduler** to schedule to run in **Grid Nodes**. An **Agent Runtime Environment** is installed in each **Grid Node** to allow the code of agents to be executed.

3.1 The Simulation Process

In this paper, we assume that the agents only interact locally with the environment. That means an agent can only see and influence the small surrounding area of the point where it locates. There is no interaction between agents.

We use the time discrete model for the simulation. Supposed that at time i , there are N Grid Nodes running simultaneously. The **Agent Application Center** divides the environment into N areas. It then creates N jobs comprising agents in these N areas and sends to the **Grid Layer**. The **Grid Scheduler** will schedule the jobs for the N **Grid Nodes**. At each **Grid Node**, the **Agent Runtime Environment** is initialized and the code of assigned agents is run for the time i . After finishing, the state of the area as well as the state of each agent will be returned to the **Agent Application Center**. The **Agent Application Center** waits until all N results returned. Then it collects data and analysis the environment again. There will be agents that leave an area to move to the

new area. They will be removed from the old corresponding **Grid Node** and transferred to the new corresponding **Grid Node**. If there are nodes joining (or leaving) the Grid, the **Agent Application Center** needs to re-divide the environment.

3.2 Dividing the Agents into Groups

In a normal agent based simulation platform, each agent is updated with the status of the whole environment. However, this approach is not suitable for simulation on a grid because it requires highly intensive communication to update the whole environment to every grid nodes, which might create a painful bottleneck at the server. To overcome this problem, we divide the whole environment dynamically into areas. For load-balancing, the number of agent in each area need to be roughly equal. The division should also be simple to be executed very fast in the Agent Application Server. Suppose that the environment has a geometrical shape of a rectangle. We now have to divide K agents into N groups. We split the map vertically, so that each area is a rectangle contains roughly K/N agents (see Figure 3). In this way, we want to minimize the number of agent moving to new areas in each time step.

To ensure load-balancing perfectly, after each round, the division should be done again. However, this takes time and may change the area of agents unnecessarily. We therefore only re-divide the environment in two cases. The first case is when there are nodes joining (or leaving) the Grid - as a result, the number of area changes. The second case is when the maximum difference between the number of agent in each area is larger than a defined threshold. The change

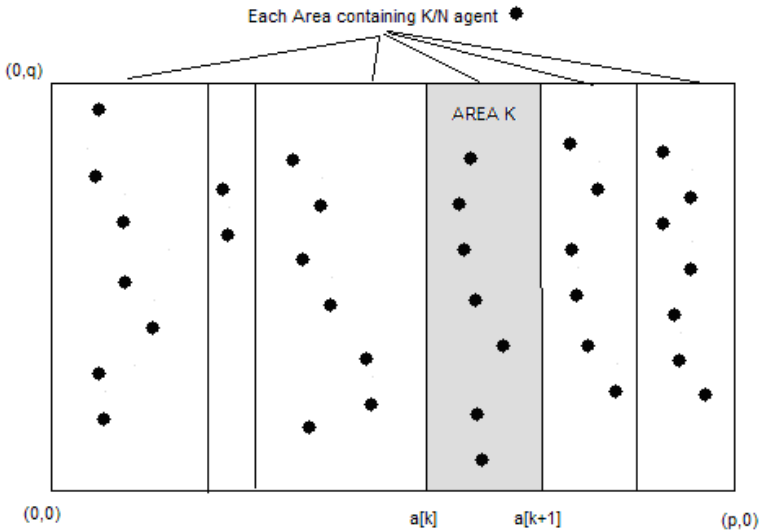


Fig. 3. Dividing the agents into groups in a location based manner

in number of area as well as the information about agent changing areas are managed by the Agent Area Managers in order to decide whether to re-divide or not.

4 Experiments and Results

The problem we use to carry out the simulation experiments is the so-called *central nest foraging* [12] for an ant colony, and it consists of two main phases: starting from the nest and exploring for food, then starting from the food source to carry the food back to the nest. The problem simulation is done based on the pheromone based utility model proposed in [11].

The problem is described as follows: the environment is a nontoroidal grid world and consists of a nest location and some N food source locations. Ants leave the nest in search of the food sources. From any given square, an ant may move to any eight-neighbor square which is not occupied by too many ants or does not contain an obstacle. When it happens upon a food source, the ant becomes laden with food. When reaching the nest again, the ant leaves the food at the nest and begins searching again. The goal is to maximize the rate of food brought to the nest from the food sources.

We have carried the experiment using 1, 2, 4, and 8 Grid nodes each with the same specification of Pentium IV 2400 MHz processor and 1024 MB of memory and running Windows XP operating system. All these nodes were interconnected over a shared student laboratory LAN network of 100 Mbps. The Agent Application Center with the Grid Server was installed on a separate computer. We have run the simulation on a map of 7000x7000 for 100 time step and the time (measured by minutes) taken for each experiment is shown in Table II. As can be seen from this Table, the framework allows the pretty good scalability when the number of agent increases as well as when the grid nodes increases.

To estimate the communication overheads between Grid Nodes and the Grid Server, we measure the number of bytes that was sent between Grid Nodes and the Grid Server during the simulation. The result is shown in Figure 4. From the figure, we can see that the more Grid Nodes joining the experiments, the more

Table 1. The time to run our simulation with 5000, 10000, 20000, 40000, 100000 and 200000 ants on 1, 2,4 and 8 computers (in minutes)

Number of ant	1 computer	2 computers	4 computers	8 computers
5000	24.53	12.921	6.531	3.200
10000	24.671	12.984	6.593	3.100
20000	25.000	13.265	6.409	3.205
40000	25.500	14.078	7.015	3.500
100000	27.2187	15.986	7.859	3.800
200000	29.9687	19.093	9.015	4.615

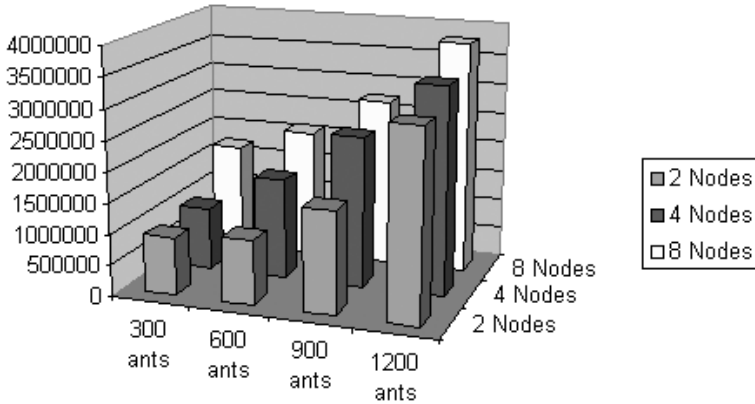


Fig. 4. The communication overheads between Grid Nodes and Grid Server (in bytes) for different number of Grid nodes and different number of ants

overhead are sent. When there is a node joining the experiment, the map will be re-divided into more areas. As a result, the number of ant moving between areas may increase. However, the communication overhead only increases linearly when increasing the number of nodes taking part in the experiments.

5 Conclusion

Different from multi-agent based simulation on a single computer, on the Grid, the agents are separated in different machines. The way the agents are separated decides the amount of network communication among the Grid system. Taking into account this, we have presented our framework for a desktop grid platform based on BOINC for multi-agent based simulation. We have proposed an approach to divide the agents to send to grid nodes so that we can keep load balancing for the distributed simulation while optimizing the communication between grid nodes and the grid server. We have implemented the food foraging simulation to evaluate the feasibility of the framework. The experiments show that the system scales pretty well when the number of agent increases as well as when the number of grid node increases.

References

- [1] Anderson, D.P.: Boinc: A system for public-resource computing and storage. In: 5th IEEE/ACM International Workshop on Grid Computing (2004)
- [2] Arikawa, H., Murata, T.: Implementation issues in a grid-based multi-agent simulation system used for increasing labor supply. *Review of Socionetwork Strategies* 1(1), 1–13 (2007)
- [3] Axelrod, R.: Advancing the art of simulation in the social sciences. *Complexity* 3(2), 193–199 (1997)

- [4] Axtell, R., Epstein, J.M.: *Growing Artificial Societies: Social Science From the Bottom Up*. MIT Press, Cambridge (1996)
- [5] Berry, B.J.L., Kiel, L.D., Elliot, E.: Adaptive agents, intelligence, and emergent human organization: Capturing complexity through agent-based modeling. *Proceedings of the National Academy of Sciences* 1999 (suppl. 3), 7178–7188 (2002)
- [6] Gasser, L., Kakugawa, K., Chee, B., Esteva, M.: Smooth scaling ahead: Progressive mas simulation from single pcs to grids. In: *Proceedings of the Joint Workshop on Multi-Agent & Multi-Agent-Based Simulation, Autonomous Agents & Multi Agent Systems (AAMAS)*, pp. 1–10 (2004)
- [7] Gazendam, H.W.M.: Theories about architectures and performance of multi-agent systems. In: *III European Congress of Psychology* (1993)
- [8] Gilbert, N., Troitzsch, K. (eds.): *Simulation for the Social Scientist*. Open University Press, Buckingham (1999)
- [9] Kim, C.H., Lee, T.D., Hwang, S.C., Jeong, C.S.: Grid-based parallel and distributed simulation environment. In: Malyshkin, V.E. (ed.) *PaCT 2003*. LNCS, vol. 2763, pp. 503–508. Springer, Heidelberg (2003)
- [10] Mengistu, D., Lundberg, L., Davidsson, P.: Performance prediction of multi-agent based simulation applications on the grid. In: *Proceedings of WASET 1921* (2007)
- [11] Panait, L., Luke, S.: A pheromone-based utility model for collaborative foraging. In: *The Third International Joint Conference on Autonomous Agents and Mutli-Agent Systems* (2004)
- [12] Sudd, J.H., Franks, N.R.: *The Behavioral Ecology of Ants*. Chapman & Hall, New York (1987)
- [13] Timm, I.J., Pawłuszczuk, D.: Large scale multiagent simulation on the grid. In: *Proceedings of the Workshop on Agent-based Grid Economics (AGE 2005)* (2005)
- [14] Zhang, C., Liu, Y., Zhang, T., Zha, Y.: Integration of the distributed simulation into the OGSA model. In: Li, M., Sun, X.-H., Deng, Q.-n., Ni, J. (eds.) *GCC 2003*. LNCS, vol. 3032, pp. 200–204. Springer, Heidelberg (2004)

An Interest Rate Adjusting Method with Bayesian Estimation in Social Lending

Masashi Iwakami¹ and Takayuki Ito^{1,2}

¹ Nagoya Institute of Technology, Gokiso-cho Nagoya 4668555, Japan

² Massachusetts Institute of Technology, Cambridge, MA 02142 USA

`iwakami@itolab.mta.nitech.ac.jp`,

`ito.takayuki@nitech.ac.jp`

Abstract. In social lending, in which an individual lends or borrows money using an SNS network, a person who lends money must take a risk that the money won't be returned. Since social lending is a comparatively new field, very few studies have been made. Therefore, we present an experimental assessment of the influence of the updating of an interest rate using Bayesian estimation, which takes into consideration the influence of groups with agents. Our method decreases dispersions of the delay of the borrower in payment with the increasing loan history of the borrower. As a result, when the lenders are risk-averse (risk means the dispersions of the delay of the borrower at each interest rate), the number of transactions increases. Therefore, our method is effective because it can cause the transactions of lenders who are risk-averse to increase.

Keywords: Bayesian Estimation, Social Lending.

1 Introduction

In recent years, social lending has been spreading. Social lending is, in its broadest sense, a certain breed of financial transaction (primarily lending and borrowing) that occurs directly on the Internet between individuals without the intermediation/participation of a traditional financial institution. The followings are the reasons social lending is spreading.

1. For a lender, lending money through social lending gives a higher interest rate than depositing with a deposit company.
2. For a borrower, borrowing money through the social lending gives a lower interest rate than borrowing money from a loan company.
3. For a borrower, it is easy to pass the loan examination.
4. There is a sense of security based on the information disclosure of SNS.

The reason for (1) and (2) is that an unnecessary fee such as a brokerage fee is not required. The reason for (3) is that there are many lenders. A bank is only one lender, whereas in social lending there are many lenders. The reason

for (4) is that the past history and preferences of users can be seen in the SNS community. With increasing information about the borrower, it will be easy for the lender to judge the borrower.

In the last few years, the number of social lending companies has increased. For example, the following companies are famous social lending companies: PROSPER[8], Lending Club[9], Zopa[10]. Each companies has unique attributes. PROSPER adopts an auction procedure for the interest rate decision of the borrower. The lender who bids for the smallest interest rate for a borrower is selected as an agency to lend money. Lending Club decides the interest rate of the borrower based on the FICO score, etc. This company has the feature that it matches lenders and borrowers with its original development engine. Zopa runs a business in two or more countries and there is a difference in the interest rate decision method in every country. Zopa of USA makes the decision on the interest rate and the examination of the financing in cooperation with a credit cooperative. The capital of lenders is put in a certificate of deposit that the federal government guarantees. Thus, it has the feature that the invested money will certainly be returned with interest even if payment failure occurs.

In social lending, lenders must bear risk. This is because the default risk in the financing moves from a loan company such as a bank to the lenders because of lending and borrowing money among individuals. Therefore, it is a subject of interest as to how a management site determines the interest rate corresponding to the borrower's risk. Since this social lending is a comparatively new field, very few studies have been made. Therefore, in this paper, we present a preliminary experimental result of the influence of the updating of an interest rate using Bayesian estimation, which takes into consideration the influence of groups with agents. The goal of the method this article suggests is to minimize risk. With Bayesian estimation, the accuracy of the interest rate presumption improves with the increasing loan history of the borrower. There are some definitions of the risk; we define the condition (dispersion) of the return rate in each rate as a risk.

Social lending mechanisms are one of the newest electronic markets. Researchers in multi-agent systems [Weiss,2000] and computational mechanism designs [Nisan,2008;Cramton,2006;Dash,2003] are focusing on building an efficient mechanism or a rule for self-interested agents who have private information. A social lending mechanism is such a mechanism for lending and borrowing money among self-interested agents. However, there have been no such studies on social lending mechanisms because they are quite new. Thus, in this paper, we focus on a new method for adjusting interest rates as a first step in social lending mechanisms, which has not been proposed or studied yet.

The remainder of the paper is organized as follows. First, we propose a new model of the interest rate decision model used in this dissertation. Second, we describe the outline of the simulation by the agent. Third, we present an experimental assessment of this model. Finally, we conclude with a discussion of possible avenues for future work.

2 The Interest Rate Decision Model

2.1 Bayesian Estimation Considering Influence of Group

In Bayes' theorem, the observed phenomenon is used and posterior probability distribution is updated by a likelihood. The shape of the posterior probability distribution is different depending on the setting of the likelihood. In this paper, we propose a method of adding the payment delay probability distributions of the group to which a borrower belongs to a likelihood at a fixed rate.

The reason to add the probability distribution of the group to the likelihood is based on comprehensive thinking. For instance, let's assume a scene in which you judge a student's scholastic achievements. At this time, assuming that student A of the highest grade class got 0 points on a 100-point test, the scholastic ability of this student A is 0 points if you judge his ability only from the individual test result. However, it is rare for a student of the highest grade class to get 0 points. It seems to be that there was some accident. Therefore, he is judged taking into consideration the average marks of the students of his class. You can give mark to A take the fact that he is the student of the highest grade class into account. Taking the influence of the group into consideration provides a more realistic assessment of student A.

Let us, for the moment, call the probability that a borrower is behind on a payment probability π . Also, X is defined to be 1, 0, depending on whether there is a delay or no delay.

$$X = \begin{cases} 1 & (\text{delay}) \\ 0 & (\text{no delay}) \end{cases}$$

The delay probability is π . The probability distribution of X is defined as follows.

$$Pr\{X = x\} = \begin{cases} \pi & (x = 1) \\ 1 - \pi & (x = 0) \end{cases} \tag{1}$$

X is a random variable. x is a realization value. Also, $p(x|\pi)$ is defined as follows.

$$p(x|\pi) = Pr\{X = x\}$$

It follows that $p(x|\pi)$ is the probability function of X . This probability function $p(x|\pi)$ is equivalent to the likelihood of the borrower in the Bayes' theorem. In our method, we apply the likelihood as follows.

$$l(\pi|x) = p(x|\pi) \cdot Pr_{group} \tag{2}$$

This is the likelihood of the borrower multiplied by the payment delay probability distributions of the groups to which a borrower belongs.

When a borrower belongs to N groups, it is a natural idea to give a priority to the groups and take the weighted mean of those probability distribution. Therefore, Pr_{group} is defined as follows.

$$Pr_{group} = \sum_{k=1}^N w_k Pr_k \quad \left(\sum_{k=1}^N w_k = 1 \right) \tag{3}$$

Here, w_k is the weight for each probability of the groups.

3 Interest Rate Correction of the Borrower with Bayesian Estimation

The steps for modifying interest of borrowers with the likelihood consists of the following four steps.

[Step 1: Calculation of posterior distribution] Posterior distribution is calculated from the borrower's repayment history by using the likelihood of the (2). The calculating formula is as follows.

$$p(\pi|x) = K \cdot l(\pi|x)p(\pi) \quad (4)$$

x is an observed event. $p(\pi)$ is a prior distribution. K is a standardization fixed number. In the calculating formula of (4), K is the value to satisfy expression (5).

$$\int_0^1 K \cdot l(\pi|x)p(\pi)d\pi = 1 \quad (5)$$

[Step 2: Calculation of the predictive delay probability] The delay probability is calculated by using the posterior distribution calculated in Step 1. The predictive delay probability is the expectation of the posterior distribution. Therefore, the desired delay probability *Prob* is defined as follows.

$$Prob = \int_0^1 \pi p(\pi|x)d\pi \quad (6)$$

[Step 3: Calculation of the proper rate] On the basis of the predictive delay probability that was calculated in Step 2, the proper interest rate is calculated. Since there is an average delay probability for every interest rate, the interest rate of the nearest delay probability is selected from among them.

[Step 4: Adjustment of the rate] The interest rate requested in Step 3 is compared with the present interest rate of the borrower. When the interest rate that was calculated in Step 3 is much different from the present interest rate, the interest rate is corrected. Concretely, when the interest rate requested in Step 3 is larger than the present interest rate by one interest rate unit or more, the interest rate is raised by one interest rate unit. When the interest rate requested in Step 3 smaller than the present interest rate by one interest rate unit or more, the interest rate is lowered by one interest rate unit. The following illustrates the interest rate adjustment algorithm shown in 3.

```

1. procedure rate_adjustment(BorrowerAgent)
2.   post = PostDistribution(BorrowerAgent)
3.   predictedProb = PredictProbability(post)
4.   properInterest = GetProperInterest(predictedProb)
5.   step = 0.1
6.   if properInterest > currentInterest + step
7.     IncreaseInterest(BorrowerAgent)
8.   elseif currentInterest - step > properInterest
9.     DecreaseInterest(BorrowerAgent)
10.  end
11.  return

```

The range of the interest rate applied to this experiment is assumed to be ten stages for simplification. The width of the interest rate is assumed to be 1% unit. The minimum interest rate is 1%, and then the next interest rate is 2%, 3%, ..., 10%. The maximum interest rate is 10%. As the return delay probability of the borrower increases, the interest rate of the borrower increases. For instance, 1% is the interest rate of the borrower whose delay probability is 10%. 2% is the interest rate of the borrower whose delay probability is 20%.

4 Experiments

4.1 Agent Simulation

In this paper, we conducted agent simulations to see how the rate adjustment mechanism works. Concretely, plural borrower agents and plural lender agents are interacted and matched in simulations. By matching processing, we examine whether the lender agent finances the borrower agent. The borrower agent has a random delay probability, and generates the payment delay according to the probability. The lender agent confirms the borrower agent's repayment history when matching it, and decides whether or not it can finance the borrower according to each utility function (the function that decides whether or not it can finance the borrower). We expected that the interest rate adjustment mechanism functions well when the number of transactions is many.

There are three main types of lenders: a risk-averse-type agent, a risk-neutral-type agent, and a risk-seeking-type agent. Each type is different in risk refusal coefficient λ in the (7). For instance, the risk-averse type has $\lambda = 1$, the risk-neutral type has $\lambda = 0$, and the risk-seeking type has $\lambda = -1$. The risk refusal coefficient is a coefficient to affect dispersion of the delay probability of plural agents at every rate.

In the field of financial engineering [1], it is thought that a brand that has a large dispersion such as profit dispersion is high-risk. When thinking about the delay frequency of the borrower agent in each interest rate as profit, it will be the same as the brand of the investment. The distribution of the delay frequency of each interest rate is considered as a risk. By the way, the utility function of the lender agent is defined as follows.

$$U = R - \lambda\sigma^2 \quad (7)$$

R is an expectation return, λ is a risk refusal coefficient, and σ^2 is a dispersion of the delay frequency.

4.2 Setting

In this experiment, the borrower agent who held the delay probability generated at random inside is generated. Ten payments are carried out respectively, and repayment is repeated 20 times. The lender agents judge whether or not they can lend money to the borrower based on their utility functions, whenever ten

repayments end. In the actual experiment, the operation mentioned above is repeated 100 times. The average of 100 operations is adopted as the result.

The parameters in this experiment are as follows.

- Number of borrower agents: 100
- Number of lender agents: 100
- Interest rate width: $0.01 \cdot 0.10$ (10 step)
- Delay probability of borrower agent: $\{0, 0.1, 0.2, \dots, 0.9\}$
- Risk refusal coefficient of lender: $\{-1, 0, 1\}$

In our experiments, we ran 100 negotiations in every condition. Our code was implemented in MATLAB R2008a.

4.3 Experimental Results

Figure 1 shows two graphs when the interest rate is adjusted and when the interest rate is not adjusted. In the graph of Figure 1, the sum of dispersions of the repayment delay rate of the agent at each interest rate was used as a comparison method. In the method for not adjusting the interest rate, the change was not seen in the sum of dispersions though the number of dealings increased. In the method for adjusting the interest rate, the result of the sum of dispersions of each interest rate decreased as the number of dealings increased. The reason the sum of dispersions decreases is that the repayment delay rate of the set interest rate approaches the repayment delay rate of the borrower. As mentioned above, with interest rate correction of a borrower using Bayes' estimation, the

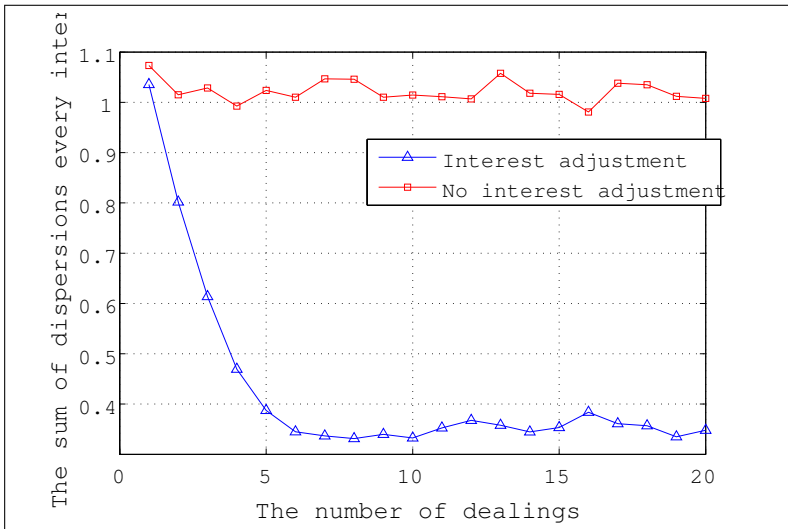


Fig. 1. Dispersion

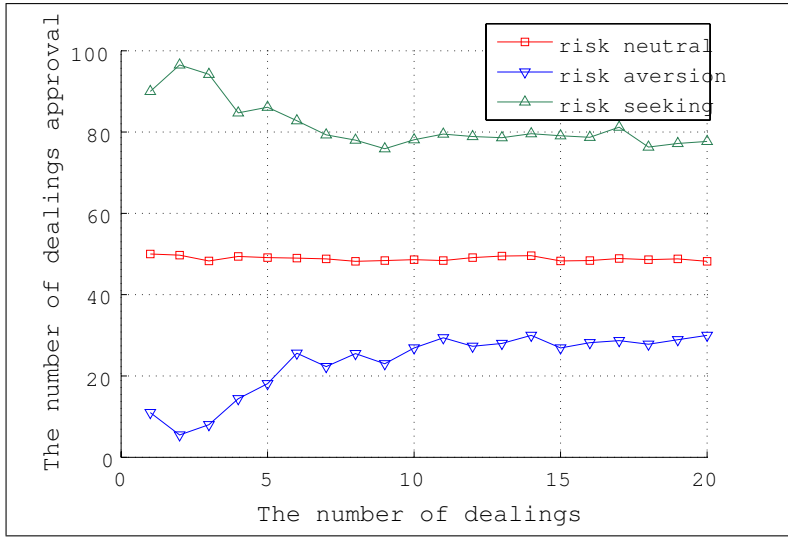


Fig. 2. Number of dealings

dispersions of the repayment delay rate of the borrower for each interest rate decrease.

Figure 2 shows the comparison of the numbers of dealings approved. These are the graphs of a risk-seeking type, a risk-neutral type, and a risk-averse type respectively. The number of dealings approved decreased for the risk-seeking type agent whenever the number of dealings increased. On the other hand, the number of dealings approved increased for the risk-averse type agent whenever the number of dealings increased. The change was not seen for the risk-neutral-type agent in the number of dealings approved. The reason for these results is that the dispersion of each interest rate decreases whenever the number of dealings increases.

It turned out that it is possible to increase the number of dealings to a risk-averse-type agent by making an interest rate correction of a borrower using Bayes' estimation. In general, people tend to be risk-averse. Thus, this result implies a possibility that our interest rate correction method could increase the number of dealings. Future work include more detailed comparisons between the case of risk-neutral agents and the case of risk-averse agents. Also, mixing the types could cause the number of dealings approved.

5 Conclusions

In this paper, we proposed a rate update algorithm using the Bayes' update in the field of social lending. With this method, small interest rate dispersion is attained with the increasing loan history of the borrower. As a result, when the

lender is risk-averse, the number of transactions increases. Therefore, our method is effective because it can cause the transactions of lenders who are risk-averse to increase. A topic for future research is proposal of a method of deciding the proper interest rate even when the amount of data is small.

References

1. Luenberger, D.G.: Investment Science. Oxford University Press, Oxford (1997)
2. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge (2000)
3. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, Englewood Cliffs (2002)
4. Dash, R.K., Jennings, N.R., Parks, D.C.: Computational-Mechanism Design: A Call to Arms. IEEE Intelligent Systems 18, 40–47 (2003)
5. Cramton, P., Shoham, Y., Steinberg, R.: Combinatorial Auctions. MIT Press, Cambridge (2006)
6. Bolstad, W.M.: Introduction to Bayesian Statistics. Wiley Interscience, Hoboken (2007)
7. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
8. Prosper, <http://www.prosper.com/>
9. LendingClub, <http://www.lendingclub.com/home.action>
10. Zopa, <https://us.zopa.com/>

A Temporal Logic for Stochastic Multi-Agent Systems

Wojciech Jamroga

Department of Informatics, Clausthal University of Technology, Germany
wjamroga@in.tu-clausthal.de

Abstract. Typical analysis of Markovian models of processes refers only to the expected utility that can be obtained by the process. On the other hand, modal logic offers a systematic method of characterizing processes by combining various modal operators. A multivalued temporal logic for Markov chains and Markov decision processes has been recently proposed in [1]. Here, we discuss how it can be extended to the multi-agent case. We relate the resulting logic to existing (two-valued) logics of strategic ability, and present fixpoint characterizations for some natural combinations of strategic and temporal operators.

Keywords: Temporal logic, multi-agent system, Markov decision process.

1 Introduction

There are many different models of agents and multi-agent systems; however, most of them follow a similar pattern. First of all, they include information about possible situations (states of the system) that defines relations between states and their external characteristics (essentially, “facts of life” that are true in these states). Second, they provide information about relationships between states (e.g., possible transitions between states). Models that share this structure can be, roughly speaking, divided into two classes. *Qualitative models* provide no numerical measures for these relationships. *Quantitative models* assume that relationships are measurable, and provide numerical information about the degrees of relations. In [1], we explored analogies between transition systems and Markovian models in order to provide a more expressive language for reasoning about, and specification of agents in stochastic environments. In [2], we tentatively extended the framework to the multi-agent case. Here, we present some formal results on the multi-agent version of the language.

Analysis of quantitative process models is usually based on the notion of expected reward. On the other hand, logical approaches are most often concerned with “limit properties” like the existence of an execution path that displays a specific temporal pattern. We believe that both kinds of properties are interesting and worth using to describe processes. For instance, besides the expected value of cumulative future reward, we can ask of the maximal (or minimal) cumulative reward. Or, we might be concerned with the expected value of minimal guaranteed reward etc. A typical analysis of multi-agent Markov decision processes is

even more constrained, as we assume that all the agents in the system cooperate to achieve a common goal (i.e., maximize their common expected cumulative reward). Our extension allows to study the outcomes that can be obtained by *various* groups of agents.

The roots of our proposal can be traced back to multivalued logics on one hand (e.g., fuzzy logics [3] and probabilistic logics [4,5]), and (crisp) modal logics of probability [6,7,8] on the other. A closer inspiration comes from multi-valued modal logics [9,10,11,12,13]. Of the latter, [11,12,13] are particularly relevant, as they define multi-valued versions of temporal logic. Still, the version of Markov Temporal Logic proposed here is (to our best knowledge) the first multivalued logic for reasoning about strategic abilities of agents in stochastic multi-agent systems.

We begin by recalling the basic idea of Markov Temporal Logic (MTL) from [1] (Section 2). The remaining sections present the original contribution of the paper: the syntax and semantics of the multi-agent MTL was only presented at a workshop with informal proceedings [2], and the theoretical results (relationship to ATL*, fixpoint properties) are entirely new.

2 Markov Temporal Logic

In this section we recall the idea of Markov Temporal Logic (MTL) from [1]. The logic allows for flexible reasoning about outcomes of agents acting in stochastic environments. The core of the logic is called MTL_0 , and addresses outcomes of Markov chains. Intuitively, MTL_0 can be seen as a quantitative analogue of the branching-time logic CTL^* [14].

2.1 Basic Models: Markov Chains

Typically, a Markov chain [15,16] is a directed graph with probabilistic transition relation. In our definition, we include also a device for assigning states with utilities and/or propositional values. This is done through *utility fluents* which generalize atomic propositions in modal logic in the sense that they can take both numerical and qualitative truth values.

Definition 1 (Domain of truth values). A domain $D = \langle U, \top, \perp, \neg \rangle$ consists of: (1) a set $U \subseteq \mathbb{R}$ of utility values (or simply utilities); (2) special values \top, \perp standing for the logical truth and falsity, respectively; $\hat{U} = U \cup \{\top, \perp\}$ will be called the extended utility set; and, finally, (3) a complement function $\neg : \hat{U} \rightarrow \hat{U}$. A domain should satisfy the conditions specified in [7], omitted here for lack of space.

Definition 2 (Markov chain). A Markov chain over domain $D = \langle U, \top, \perp, \neg \rangle$, and a set of utility fluents Π is a tuple $M = \langle St, \tau, \pi \rangle$, where:

- St is a set of states (we will assume that the set is finite and nonempty throughout the rest of the paper);

- $\tau : St \times St \rightarrow [0, 1]$ is a stochastic transition relation that assigns each pair of states q_1, q_2 with a probability $\tau(q_1, q_2)$ that, if the system is in q_1 , it will change its state to q_2 in the next moment. For every $q_1 \in St$, $\tau(q_1, \cdot)$ is assumed to be a probability distribution, i.e. $\sum_{q \in St} \tau(q_1, q) = 1$.
By abuse of notation, we will sometimes write $\tau(q)$ to denote the set of states accessible in one step from q , i.e. $\{q' \mid \tau(q, q') > 0\}$.
- $\pi : \Pi \times St \rightarrow \hat{U}$ is a valuation of utility fluents.

A run in Markov chain M is an infinite sequence of states $q_0q_1 \dots$ such that each q_{i+1} can follow q_i with a non-zero probability. The set of runs starting from state q is denoted by $\mathcal{R}_M(q)$.¹ Let $\lambda = q_0q_1 \dots$ be a run and $i \in \mathbb{N}_0$. Then: $\lambda[i] = q_i$ denotes the i th position in λ , and $\lambda[i..\infty] = q_iq_{i+1} \dots$ denotes the infinite subpath of λ from position i on.

2.2 Logical Operators as Minimizers and Maximizers

Note that – when truth values represent utility of an agent – temporal operators “sometime” and “always” have a very natural interpretation. “Sometime p ” ($\diamond p$) can be rephrased as “ p is achievable in the future”. Thus, under the assumption that agents want to obtain as much utility as possible, it is natural to view the operator as maximizing the utility value along a given temporal path. Similarly, “always p ” ($\square p$) can be rephrased as “ p is guaranteed from now on”. In other words, $\square p$ asks for the minimal value of p on the path. On a more general level, every universal quantifier is essentially a minimizer of truth values, while existential quantifiers can be seen as maximizers. Thus, $E\gamma$ (“there is a path such that γ ”) maximizes the utility specified by γ across all paths that can occur; likewise, $A\gamma$ (“for all paths γ ”) minimizes the value of γ across paths. Also, disjunction and conjunction can be seen as a maximizer and a minimizer: $\varphi \vee \psi$ reads easily as “the utility that can be achieved through φ or ψ ”, while $\varphi \wedge \psi$ reads as “utility guaranteed by both φ and ψ ”.

2.3 MTL₀: A Logic of Markov Chains

Operators of MTL_0 include path quantifiers E, A, M for the maximal, minimal, and average outcome of a set of temporal paths, respectively, and temporal operators \diamond, \square, m for the maximal, minimal, and average outcome along a given path.² Propositional operators follow the same pattern: \vee, \wedge, \oplus refer to maximization, minimization, and weighted average of outcomes obtained from different utility channels or related to different goals. Finally, we have the “defuzzification” operator \preceq , which provides a two-valued interface to the logic. $\varphi_1 \preceq \varphi_2$ yields “true” if the outcome of φ_1 is less or equal to φ_2 , and “false” otherwise. Among other advantages, it allows to define the classical computational problems of validity, satisfiability and model checking for MTL.

¹ If the model is clear from the context, the subscripts will be omitted.

² We allow to discount future outcomes with a discount factor c . Also, we introduce the “until” operator \mathcal{U} , which is more general than \diamond .

Let $Bool(\omega) = \neg\omega \mid \omega \wedge \omega \mid \omega \oplus_c \omega \mid \omega \preceq \omega$ denote quasi-Boolean combinations of formulae of type ω . The syntax of MTL_0 can be defined by the following production rules:

$$\begin{aligned}\varphi &::= p \mid Bool(\varphi) \mid E\gamma \mid M\gamma, \\ \gamma &::= \varphi \mid Bool(\gamma) \mid \bigcirc_c \gamma \mid \square_c \gamma \mid \gamma \mathcal{U}_c \gamma \mid \mathbf{m}_c \gamma,\end{aligned}$$

where $p \in \Pi$ is a utility fluent, and c is a discount factor such that $0 < c \leq 1$. Additionally, we define $\varphi_1 \cong \varphi_2 \equiv (\varphi_1 \preceq \varphi_2) \wedge (\varphi_2 \preceq \varphi_1)$. Boolean constants \top, F (“true”, “false”), disjunction, and the “sometime” temporal operator \diamond are defined in the standard way. The following shorthands are used for discount-free versions of temporal operators: $\bigcirc \equiv \bigcirc_1, \diamond \equiv \diamond_1, \square \equiv \square_1, \mathcal{U} \equiv \mathcal{U}_1$.

Example 1. Let r be a utility fluent that represents the immediate reward at each state. The following MTL_0 formulae define some interesting characteristics of a process: $\text{Mm}_{0.9}r$ (expected average reward with time discount 0.9), $\text{Am}_{0.9}r$ (guaranteed average reward with the same discount factor), $\text{M}\square r$ (expected minimal undiscounted reward), and $\text{A}\diamond r$ (guaranteed maximal reward).

The main idea behind MTL_0 is that formulae can refer to both quantitative utilities and qualitative truth values. Thus, we treat complex formulae as fluents, just like the atomic utility fluents from Π , through a valuation function that assigns formulae with extended utility values from \hat{U} . Let $M = \langle St, \tau, \pi \rangle$ be a Markov chain over domain $D = \langle U, \top, \perp, \neg \rangle$ and a set of utility fluents Π . The valuation function $[\cdot]$ is defined below.

- $[p]_{M,q} = \pi(p, q)$, for $p \in \Pi$;
- $[\neg\varphi]_{M,q} = \overline{[\varphi]_{M,q}}$;
- $[\varphi_1 \wedge \varphi_2]_{M,q} = \min([\varphi_1]_{M,q}, [\varphi_2]_{M,q})$;
- $[\varphi_1 \oplus_c \varphi_2]_{M,q} = (1 - c) \cdot [\varphi_1]_{M,q} + c \cdot [\varphi_2]_{M,q}$;
- $[\varphi_1 \preceq \varphi_2]_{M,q} = \top$ if $[\varphi_1]_{M,q} \leq [\varphi_2]_{M,q}$ and \perp else;
- $[E\gamma]_{M,q} = \sup\{[\gamma]_{M,\lambda} \mid \lambda \in \mathcal{R}(q)\}$;
- The Markovian path quantifier $M\gamma$ produces the expected truth value γ across all the possible runs, cf. [16] for the formal construction;
- $[\varphi]_{M,\lambda} = [\varphi]_{M,\lambda[0]}$;
- $[\neg\gamma]_{M,\lambda}, [\gamma_1 \wedge \gamma_2]_{M,\lambda}, [\gamma_1 \oplus_c \gamma_2]_{M,\lambda}, [\gamma_1 \preceq \gamma_2]_{M,\lambda}$: analogous to Boolean combinations of “state formulae” φ ;
- $[\bigcirc_c \gamma]_{M,\lambda} = c \cdot [\gamma]_{M,\lambda[1..\infty]}$;
- $[\square_c \gamma]_{M,\lambda} = \inf_{i=0,1,\dots} \{c^i [\gamma]_{M,\lambda[i..\infty]}\}$;
- $[\gamma_1 \mathcal{U}_c \gamma_2]_{M,\lambda} = \sup_{i=0,1,\dots} \{ \min(\min_{0 \leq j < i} \{c^j [\gamma_1]_{M,\lambda[j..\infty]}\}, c^i [\gamma_2]_{M,\lambda[i..\infty]}) \}$;
- The Markovian temporal operator \mathbf{m}_c produces the average discounted reward along the given run:

$$[\mathbf{m}_c \gamma]_{M,\lambda} = \begin{cases} (1-c) \sum_{i=0}^{\infty} c^i [\gamma]_{M,\lambda[i..\infty]} & \text{if } c < 1 \\ \frac{\limsup_{i \rightarrow \infty} \frac{1}{i+1} \sum_{j=0}^i [\gamma]_{M,\lambda[j..\infty]} + \liminf_{i \rightarrow \infty} \frac{1}{i+1} \sum_{j=0}^i [\gamma]_{M,\lambda[j..\infty]}}{2} & \text{if } c = 1 \end{cases}$$

3 Reasoning about Stochastic Multi-agent Processes

Strategic abilities were already considered in MTL_1 , the version of Markov Temporal Logic for reasoning about Markov decision processes [1]. In consequence, MTL_1 can be seen as a quantitative analogue of the *single-agent* fragment of ATL^* [17] with memoryless strategies. In the more general case, a system can include multiple agents/processes, interacting with each other. To address their properties, a family of operators $\langle\langle A \rangle\rangle\varphi$ can be used, parameterized with groups of agents A . Intuitively, $\langle\langle A \rangle\rangle\varphi$ refers to how much agents A can “make out of” φ by following their best joint policy. This yields a language similar to the alternating-time temporal logic ATL^* from [17], albeit with strategic operators separated from path quantifiers.

Markov decision processes [18,19] extend Markov chains with an explicit action structure: transitions are generated by actions of an (implicit) decision maker. *Multi-agent Markov decision processes* (MMDP) [20] extend Markov decision processes to the multi-agent setting: transitions are now labeled by combinations of agents’ actions. We observe the similarity between MMDP’s and concurrent game structures which are the models of ATL^* (cf. Figure 1).

As models for our multi-agent MTL, we will use a refinement of MMDP’s similar to the version of Markov chains presented in Section 2.1. The semantics of $\langle\langle A \rangle\rangle\varphi$ is based on maximization of the value of φ with respect to A ’s joint strategies. We assume that the opponents play a strategy that minimizes φ most. This way, operator $\langle\langle A \rangle\rangle$ corresponds to the maxmin of the two-player game where A is the (collective) maximizer, and the rest of agents fills in the role of the (collective) minimizer. Note that such a semantics entails that the opponents of A must also play only *memoryless* (i.e., Markovian) strategies.

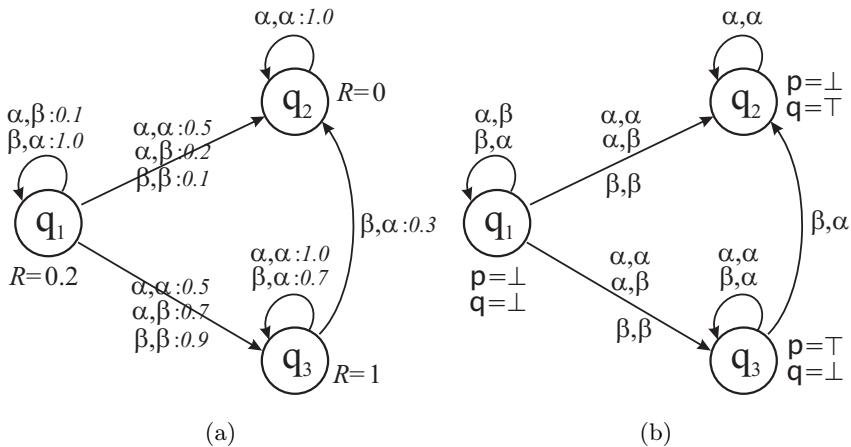


Fig. 1. (a) Simple MMDP with two agents; (b) Simple concurrent game structure

3.1 MTL₂: Syntax

Let $\mathbb{A}gt$ be the set of all agents. MTL_2 adds to MTL_0 a family of operators $\langle\langle A \rangle\rangle$, one for each group of agents $A \subseteq \mathbb{A}gt$. Formally, the syntax of MTL_2 is given by the following grammar:

$$\begin{aligned}\vartheta &::= p \mid Bool(\vartheta) \mid \langle\langle A \rangle\rangle\varphi, \\ \varphi &::= \vartheta \mid Bool(\varphi) \mid E\gamma \mid M\gamma, \\ \gamma &::= \varphi \mid Bool(\gamma) \mid \bigcirc_c \gamma \mid \square_c \gamma \mid \gamma \mathcal{U}_c \gamma \mid m_c \gamma.\end{aligned}$$

An example formula of MTL_2 is $\langle\langle 1, 2 \rangle\rangle Amr$ which makes agents 1 and 2 maximize the guaranteed average reward r with respect to their available policies.

3.2 MTL₂: Semantics

The semantics of MTL_2 is defined for a version of multi-agent Markov decision processes that incorporates qualitative as well as quantitative atomic properties of states.

Definition 3 (MMDP). *A multi-agent Markov decision process over domain $D = \langle U, \top, \perp, \neg \rangle$ and a set of utility fluents Π is a tuple $\mathcal{M} = \langle \mathbb{A}gt, St, \{Act_i\}_{i \in \mathbb{A}gt}, \tau, \pi \rangle$, where: St, π are like in a Markov chain, $\mathbb{A}gt = \{1, \dots, k\}$ is the set of agents, Act_i is the set of individual actions of agent i , and $Act = \prod_{i \in \mathbb{A}gt} Act_i$ is the space of joint actions (action profiles). $\tau : St \times Act \times St \rightarrow [0, 1]$ is a stochastic transition relation; $\tau(q_1, \alpha, q_2)$ defines the probability that, if the system is in q_1 and the agents execute α , the next state will be q_2 . For every $q \in St, \alpha \in Act$, we assume that either (1) $\tau(q, \alpha, q') = 0$ for all q' (i.e., α is not enabled in q), or (2) $\tau(q, \alpha, \cdot)$ is a probability distribution. Additionally, we define $act(q) = \{\alpha \in Act \mid \exists q'. \tau(q, \alpha, q') > 0\}$ as the set of enabled action profiles in q .*

For a joint action α , we define α^i to denote agent i 's individual part in α , and we extend the notation to sets of joint actions and agents. Also, let \mathcal{A} be a set of action profiles, and α a collective action of agents A . Then, $\mathcal{A}|\alpha = \{\beta \in \mathcal{A} \mid \beta^A = \alpha\}$ is the set of action profiles that include α .

A policy is a conditional plan that specifies future actions of an agent. Policies can be stochastic as well, thus allowing for randomness in the agent's play.

Definition 4. *An individual strategy (policy) of agent i is a function $s_i : St \times Act_i \rightarrow [0, 1]$ that assigns each state q with a probability distribution over i 's enabled actions $act(q)^i$. That is, $s(q, \alpha_i) \in [0, 1]$ for all $q \in St, \alpha_i \in act(q)^i$, and $\sum_{\alpha_i \in act(q)^i} s(q, \alpha_i) = 1$. Values of $s(q, \alpha_i)$ for $\alpha_i \notin act(q)^i$ are irrelevant. The set of all i 's strategies is denoted by Σ_i . A collective strategy s_A for team $A \subseteq \mathbb{A}gt$ is simply a tuple of individual strategies, one per agent from A . The set of all A 's collective strategies is given by $\Sigma_A = \prod_{i \in A} \Sigma_i$. The set of all strategy profiles in a model is given by $\Sigma = \Sigma_{\mathbb{A}gt}$.*

For a collective strategy s , we define s^i as the i 's individual part in s . We also extend the notation to sets of agents.

Definition 5. Policy $s \in \Sigma_A$ instantiates MMDP $\mathcal{M} = \langle \text{Agt}, St, \{Act_i\}_{i \in \text{Agt}}, \tau, \pi \rangle$ to a simpler MMDP $\mathcal{M} \dagger s = \langle \text{Agt} \setminus A, St, \{Act_i\}_{i \in \text{Agt} \setminus A}, \tau', \pi \rangle$ with

$$\tau'(q, \alpha, q') = \sum_{\alpha' \in (act(q)|\alpha)} \left(\prod_{i \in A} s^i(q, \alpha') \right) \tau(q, \alpha', q').$$

If $A = \text{Agt}$, then s instantiates \mathcal{M} to a Markov chain.

The semantics of MTL_2 extends that of MTL_0 with the following clauses:

- $[p]_{\mathcal{M},q} = \pi(p, q)$, for $p \in II$;
- $[\neg\vartheta]_{\mathcal{M},q}$, $[\vartheta_1 \wedge \vartheta_2]_{\mathcal{M},q}$, $[\vartheta_1 \oplus_c \vartheta_2]_{\mathcal{M},q}$, $[\vartheta_1 \preceq \vartheta_2]_{\mathcal{M},q}$: analogous as for “state formulae” φ ;
- $[\langle\langle A \rangle\rangle\varphi]_{\mathcal{M},q} = \sup_{s \in \Sigma_A} \inf_{t \in \Sigma_{\text{Agt} \setminus A}} \{[\varphi]_{\mathcal{M} \dagger (s,t),q}\}$;

In order to keep consistent with qualitative logics of strategic ability, we assume that instantiation of an MMDP by a policy s is “soft” in the sense that nested strategic operators discard previous instantiations and instantiate the original model again: $[\langle\langle A \rangle\rangle\varphi]_{\mathcal{M} \dagger s,q} = [\langle\langle A \rangle\rangle\varphi]_{\mathcal{M},q}$.

Example 2. Consider the multi-agent Markov decision process from Figure 1A, consisting of two agents (1 and 2). If the agents cooperate, they can maximize the expected achievable reward quite successfully, as $[\langle\langle 1, 2 \rangle\rangle M \diamond R]_{q_1} = 0.9$ (best policy: both agents play β in q_1 with probability 1; the choices at other states are irrelevant). If agent 1 is to maximize the expected achievable reward on his own, against adversary behavior of agent 2, then he is bound to be less successful: $[\langle\langle 1 \rangle\rangle M \diamond R]_{q_1} = 0.6$. Also, in this case agent 1 should employ a different policy, namely play α in q_1 with probability 1.

4 Formal Results

The semantics of MTL, presented in the previous section, portrays it as a language of arithmetic expressions that can be used to define numerical characteristics of Markov processes. However, MTL can be also seen as a *logic*, i.e. a set of *sentences* that are true in some contexts, and false (at least to a degree) in others. This view allows us to use the conceptual apparatus of mathematical logic to study e.g. the expressivity of the language. Also, we can state interesting properties of the domain (multi-agent stochastic processes) through formulae of MTL. To this end, we first define what it means for a formula to be valid and/or satisfiable.

4.1 Levels of Truth

Since every domain must include a distinguished value for the classical (complete) truth, validity of formulae can be defined in a straightforward way.

Definition 6 (Levels of validity). Let \mathcal{M} be a multi-agent Markov decision process, q a state in \mathcal{M} , and ϑ a formula of MTL_2 . Then:

- ϑ is true in \mathcal{M}, q (written $\mathcal{M}, q \models \vartheta$) iff $[\vartheta]_{\mathcal{M},q} = \top$.
- ϑ is valid in \mathcal{M} (written $\mathcal{M} \models \vartheta$) iff it is true in every state of \mathcal{M} .
- ϑ is valid for multi-agent Markov decision processes (written $\models \vartheta$) iff it is valid in every MMDP \mathcal{M} .
- Additionally, for path formulae γ , we can say that γ holds on run λ in MMDP \mathcal{M} (written $\mathcal{M}, \lambda \models \gamma$) iff $[\gamma]_{\mathcal{M},\lambda} = \top$.

The notion of validity helps to express general properties of stochastic multi-agent systems in a neat logical way. Moreover, Definition 6 allows to define the typical decision problems for MTL_2 in a natural way:

- Given a formula ϑ , the *validity problem* asks if $\models \vartheta$;
- Given a formula ϑ , the *satisfiability problem* asks if there are \mathcal{M}, q such that $\mathcal{M}, q \models \vartheta$;
- Given a model \mathcal{M} , state q and formula ϑ , the *model checking problem* asks if $\mathcal{M}, q \models \vartheta$.

For example, we can search for a model in which agent a can guarantee the average reward r to be at least 0.6 in the long run by solving the satisfiability problem for formula $0.6 \preceq \langle\langle a \rangle\rangle \text{Amr}$.

We consider model checking the most important of the three problems, since in the analysis of a stochastic system the domain specification is usually given by a procedural representation (rather than axiomatic theory). Some work on model checking multi-valued temporal logics has been reported in [11,12]. Perhaps even more importantly, computing approximate “solutions” of MDP’s is one of the central issues studied by the Markov community. Integration of the two approaches seems a very promising (and exciting) path for future research.

4.2 Concurrent Game Structures as MMDP’s. Correspondence between MTL_2 and ATL^*

Multi-agent Markov decision processes can be seen as generalizations of concurrent game structures [17], in which quantitative information is added through non-classical values of atomic statements and probabilities of transitions. Conversely, concurrent game structures can be seen as a subclass of MMDP’s with all fluents assuming only classical truth values.

Definition 7. Let \mathcal{M} be an MMDP. Formula φ is propositional in \mathcal{M} iff it can take only the values of \top, \perp , i.e., $[\varphi]_{\mathcal{M},q} \in \{\top, \perp\}$ for all $q \in \text{St}$. A concurrent game structure is an MMDP with only propositional fluents.

This way, we obtain the class of models that are used for qualitative alternating-time logics, i.e. ATL and ATL^* . Of course, when interpreting formulae of qualitative $\text{ATL} / \text{ATL}^*$, one must as well ignore the probabilities that are present in Markov decision processes. Note also that the semantics of the original $\text{ATL} / \text{ATL}^*$ uses the “history-based” notion of a strategy (i.e., strategies assign choices to *histories* rather than single states), while our MTL_2 is underpinned by a much weaker notion of *memoryless* (or positional) strategies. This makes

the two logics formally incomparable. However, we can show that MTL_2 strictly generalizes the memoryless version of ATL^* . The latter was studied in [21] under the acronym of ATL_{Ir}^* (ATL with Perfect Information and imperfect recall), and we will use the name here.

Proposition 1. *Let \mathcal{M} be a transition system, and φ a formula of ATL_{Ir}^* . Moreover, let φ' be the result of replacing every occurrence of $\langle\langle A \rangle\rangle$ with $\langle\langle A \rangle\rangle_{\text{Ir}} A$ in φ for all $A \subseteq \text{Agt}$. Then, $\mathcal{M}, q \models_{\text{MTL}_2} \varphi'$ iff $\mathcal{M}, q \models_{\text{ATL}_{\text{Ir}}^*} \varphi$.*

Proof (sketch). Let σ_A denote the set of deterministic memoryless strategies of group A .³ The proof follows by induction on the structure of φ ; here, we only sketch the induction step for the most important case, namely $\varphi \equiv \langle\langle A \rangle\rangle_{\text{Ir}} \gamma$. We recall from [21] the semantics of $\langle\langle A \rangle\rangle_{\text{Ir}}$: let $\text{out}_{\mathcal{M}}(q, s)$ be the set of paths in \mathcal{M} that can result from execution of strategy s from state q on; then, $\mathcal{M}, q \models \langle\langle A \rangle\rangle_{\text{Ir}} \gamma$ iff there is $s \in \sigma_A$ such that for every $\lambda \in \text{out}_{\mathcal{M}}(q, s)$ we have $\mathcal{M}, \lambda \models \gamma$.

“ $\text{MTL}_2 \Rightarrow \text{ATL}_{\text{Ir}}^*$ ”: Let $\mathcal{M}, q \models_{\text{MTL}_2} \langle\langle A \rangle\rangle_{\text{Ir}} A \gamma$. Then, $[\langle\langle A \rangle\rangle_{\text{Ir}} A \gamma]_{\mathcal{M}, q} = \top$, and so $\sup_{s \in \Sigma_A} \inf_{t \in \Sigma_{\text{Agt} \setminus A}} \inf_{\lambda \in \mathcal{R}_{\mathcal{M} \dagger \langle s, t \rangle}(q)} [\gamma]_{\mathcal{M} \dagger \langle s, t \rangle, \lambda} = \top$; let s^* be a strategy that maximizes the above expression. Note that all the state subformulae of γ will be in fact evaluated in the original MMDP \mathcal{M} , so we get that $\inf_{t \in \Sigma_{\text{Agt} \setminus A}} \inf_{\lambda \in \mathcal{R}_{\mathcal{M} \dagger \langle s^*, t \rangle}(q)} [\gamma]_{\mathcal{M}, \lambda} = \top$. Thus, $\forall t \in \Sigma_{\text{Agt} \setminus A} \forall \lambda \in \mathcal{R}_{\mathcal{M} \dagger \langle s^*, t \rangle}(q) [\gamma]_{\mathcal{M}, \lambda} = \top$, and by the induction hypothesis we obtain that $\forall t \in \Sigma_{\text{Agt} \setminus A} \forall \lambda \in \mathcal{R}_{\mathcal{M} \dagger \langle s^*, t \rangle}(q) \mathcal{M}, \lambda \models_{\text{ATL}_{\text{Ir}}^*} \gamma$. Now we observe that if $s \in \Sigma_A$ is a randomized strategy and $[s] \in \sigma_A$ is any determinization of s then $\mathcal{R}_{\mathcal{M} \dagger \langle [s], t \rangle}(q) \subseteq \mathcal{R}_{\mathcal{M} \dagger \langle s, t \rangle}(q)$, so also for $[s^*]$ we have that $\forall t \in \Sigma_{\text{Agt} \setminus A} \forall \lambda \in \mathcal{R}_{\mathcal{M} \dagger \langle [s^*], t \rangle}(q) \mathcal{M}, \lambda \models_{\text{ATL}_{\text{Ir}}^*} \gamma$. Finally, we take t to be the uniform randomized strategy of $\text{Agt} \setminus A$ since it does not remove any paths from the model: $\mathcal{R}_{\mathcal{M} \dagger \langle [s^*], \text{uniform} \rangle}(q) = \text{out}_{\mathcal{M}}(q, [s^*])$. In consequence, $\forall \lambda \in \text{out}_{\mathcal{M}}(q, [s^*]) \mathcal{M}, \lambda \models_{\text{ATL}_{\text{Ir}}^*} \gamma$, which concludes this part of the proof.

“ $\text{ATL}_{\text{Ir}}^* \Leftarrow \text{MTL}_2$ ”: Let $\mathcal{M}, q \models_{\text{ATL}_{\text{Ir}}^*} \langle\langle A \rangle\rangle_{\text{Ir}} \gamma$. Then, $\exists s \in \sigma_A \forall \lambda \in \text{out}_{\mathcal{M}}(q, s) \mathcal{M}, \lambda \models_{\text{ATL}_{\text{Ir}}^*} \gamma$. We take such s . By induction, $\forall \lambda \in \text{out}_{\mathcal{M}}(q, s) \mathcal{M}, \lambda \models_{\text{MTL}_2} \gamma$. Take any $t \in \Sigma_{\text{Agt} \setminus A}$, then $\mathcal{R}_{\mathcal{M} \dagger \langle s, t \rangle}(q) \subseteq \text{out}_{\mathcal{M}}(q, s)$, and hence also $\forall \lambda \in \mathcal{R}_{\mathcal{M} \dagger \langle s, t \rangle}(q) \mathcal{M}, \lambda \models_{\text{MTL}_2} \gamma$. As $\sigma_A \subseteq \Sigma_A$, we finally get that $\exists s \in \Sigma_A \forall t \in \Sigma_{\text{Agt} \setminus A} \forall \lambda \in \mathcal{R}_{\mathcal{M} \dagger \langle s, t \rangle}(q) \mathcal{M}, \lambda \models_{\text{MTL}_2} \gamma$. In consequence, $\sup_{s \in \Sigma_A} \inf_{t \in \Sigma_{\text{Agt} \setminus A}} \inf_{\lambda \in \mathcal{R}_{\mathcal{M} \dagger \langle s, t \rangle}(q)} [\gamma]_{\mathcal{M} \dagger \langle s, t \rangle, \lambda} = \top$, which concludes the proof.

Proposition 2. *There is a transition system M with states q, q' which cannot be distinguished by any formula of ATL^* nor ATL_{Ir}^* , and can be distinguished by a formula of MTL_2 .*

Proof. Consider the transition system in Figure 2, which can be seen as a concurrent game structure with a single agent ($\text{Agt} = \{1\}$) and a single action that can be executed ($\text{Act} = \{\alpha\}$). Note that states q_1, q_2 are bisimilar under CTL^* bisimulation, so the same CTL^* properties hold in both states (cf. e.g. [22]). Since the agent cannot make any real choices, both ATL^* and ATL_{Ir}^* have no more distinguishing power for this model as CTL^* , and hence the same properties of ATL^* (resp. ATL_{Ir}^*) hold in q_1, q_2 as well.

³ Recall that Σ_A is the set of all (possibly randomized) memoryless strategies of A .

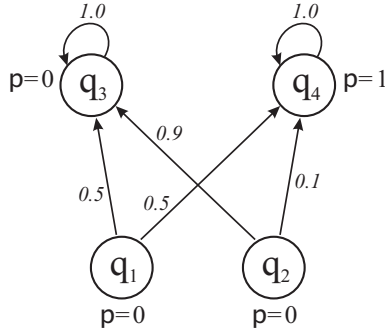


Fig. 2. MTL₂ vs. ATL*: probabilities matter!

On the other hand, we have that $[\langle\langle 1 \rangle\rangle \text{Mmp}]_{q_1} = 0.5 = [\langle\langle 1 \rangle\rangle \text{Em}_{0.5\text{p}}]_{q_1}$, and $[\langle\langle 1 \rangle\rangle \text{Mmp}]_{q_2} = 0.1 \neq 0.5 = [\langle\langle 1 \rangle\rangle \text{Em}_{0.5\text{p}}]_{q_2}$. Thus, for $\varphi \equiv (\langle\langle 1 \rangle\rangle \text{Mmp} \cong \langle\langle 1 \rangle\rangle \text{Em}_{0.5\text{p}})$, we have $q_1 \models \varphi$ and $q_2 \not\models \varphi$ (and even $q_2 \models \neg\varphi$).

The above example shows that a proper notion of bisimulation for Markov decision processes must take into account transition probabilities.

4.3 State-Based Formulae and Bellman Equations

“ATL without star” (or “vanilla ATL”) is the most often used variant of alternating-time temporal logic, mainly due to the complexity of its model checking problem and the fact that its semantics can be defined entirely in relation to states. “Vanilla” ATL can be seen as a syntactic restriction of ATL*, in which every temporal modality is preceded by exactly one path quantifier. In this section, we consider a similar syntactic restriction on MTL₂; we call it state-based MTL₂.

Definition 8. State-based MTL₂ (sMTL₂ in short) is given as follows:

$$\begin{aligned} \vartheta &::= p \mid \text{Bool}(\vartheta) \mid \langle\langle A \rangle\rangle\varphi, \\ \varphi &::= \mathbf{E}\gamma \mid \mathbf{M}\gamma, \\ \gamma &::= \mathbf{O}_c\vartheta \mid \mathbf{O}_c\vartheta \mid \vartheta\mathcal{U}_c\vartheta \mid \mathbf{m}_c\vartheta. \end{aligned}$$

Proposition 3 presents fixpoint characterizations for most modalities of sMTL₂. Note that the last validity from the list is in fact a modal formulation of *Bellman equation*, which is the basic law used in analysis of Markov decision processes. The other formulae can be seen as variants of the equation for non-standard analysis based on minimal/maximal rather than average rewards. The results from [12] suggest that $\langle\langle A \rangle\rangle \mathbf{M}\mathbf{O}_c$ and $\langle\langle A \rangle\rangle \mathbf{M}\mathcal{U}_c$ do not have fixpoint characterizations, but this remains to be formally proven.

Proposition 3. The following formulae of sMTL₂ are valid:

- $\langle\langle A \rangle\rangle \mathbf{E}\mathbf{O}_c\varphi \cong \varphi \wedge \langle\langle A \rangle\rangle \mathbf{E}\mathbf{O}_c \langle\langle A \rangle\rangle \mathbf{E}\mathbf{O}_c\varphi$;
- $\langle\langle A \rangle\rangle \mathbf{A}\mathbf{O}_c\varphi \cong \varphi \wedge \langle\langle A \rangle\rangle \mathbf{A}\mathbf{O}_c \langle\langle A \rangle\rangle \mathbf{A}\mathbf{O}_c\varphi$;

- $\langle\langle A \rangle\rangle E \varphi_1 \mathcal{U}_c \varphi_2 \cong \varphi_2 \vee \varphi_1 \wedge \langle\langle A \rangle\rangle E \bigcirc_c \langle\langle A \rangle\rangle E \varphi_1 \mathcal{U}_c \varphi_2$;
- $\langle\langle A \rangle\rangle A \varphi_1 \mathcal{U}_c \varphi_2 \cong \varphi_2 \vee \varphi_1 \wedge \langle\langle A \rangle\rangle A \bigcirc_c \langle\langle A \rangle\rangle A \varphi_1 \mathcal{U}_c \varphi_2$;
- $\langle\langle A \rangle\rangle \text{Em}_c \varphi \cong \varphi \oplus_c \langle\langle A \rangle\rangle E \bigcirc \langle\langle A \rangle\rangle \text{Em}_c \varphi$;
- $\langle\langle A \rangle\rangle \text{Am}_c \varphi \cong \varphi \oplus_c \langle\langle A \rangle\rangle A \bigcirc \langle\langle A \rangle\rangle \text{Am}_c \varphi$;
- $\langle\langle A \rangle\rangle \text{Mm}_c \varphi \cong \varphi \oplus_c \langle\langle A \rangle\rangle M \bigcirc \langle\langle A \rangle\rangle \text{Mm}_c \varphi$.

Proof (sketch). We will sketch the proof of the first validity; the others can be proved in an analogous way.

Let $L = [\langle\langle A \rangle\rangle E \square_c \varphi]_{\mathcal{M}, q}$ and $R = [\varphi \wedge \langle\langle A \rangle\rangle E \bigcirc_c \langle\langle A \rangle\rangle E \square_c \varphi]_{\mathcal{M}, q}$. It is easy to see that $R = \min([\varphi]_{\mathcal{M}, q}, c \cdot \sup_{s \in \Sigma_A} \inf_{t \in \Sigma_{\text{Agt} \setminus A}} \sup_{q' \in \tau_{\mathcal{M} \uparrow(s, t)}(q)} \sup_{s' \in \Sigma_A} \inf_{t' \in \Sigma_{\text{Agt} \setminus A}} \{[\langle\langle A \rangle\rangle E \square_c \varphi]_{\mathcal{M} \uparrow(s', t'), q'}\})$. Moreover, by [11, Proposition 8], we get that $L = \min([\varphi]_{\mathcal{M}, q}, c \cdot \sup_{s \in \Sigma_A} \inf_{t \in \Sigma_{\text{Agt} \setminus A}} \sup_{q' \in \tau_{\mathcal{M} \uparrow(s, t)}(q)} \{[\langle\langle A \rangle\rangle E \square_c \varphi]_{\mathcal{M} \uparrow(s, t), q'}\})$. Thus, in order to prove $L = R$, it is sufficient to prove that

$$\begin{aligned} & \sup_{s \in \Sigma_A} \inf_{t \in \Sigma_{\text{Agt} \setminus A}} \sup_{q' \in \tau_{\mathcal{M} \uparrow(s, t)}(q)} \{[\langle\langle A \rangle\rangle E \square_c \varphi]_{\mathcal{M} \uparrow(s, t), q'}\} \\ &= \sup_{s \in \Sigma_A} \inf_{t \in \Sigma_{\text{Agt} \setminus A}} \sup_{q' \in \tau_{\mathcal{M} \uparrow(s, t)}(q)} \sup_{s' \in \Sigma_A} \inf_{t' \in \Sigma_{\text{Agt} \setminus A}} \{[\langle\langle A \rangle\rangle E \square_c \varphi]_{\mathcal{M} \uparrow(s', t'), q'}\}. \end{aligned}$$

The difference between the sides of the equation is that in the left hand side optimal strategies s, t are chosen once (at state q), while in the right hand side strategies are re-evaluated after each step. Let s^* be a strategy of A that optimizes L , and let us take s and s' in R to be the same as s^* in L . We observe that $\inf_{t \in \Sigma_{\text{Agt} \setminus A}} \sup_{q' \in \tau_{\mathcal{M} \uparrow(s^*, t)}(q)} \{[\langle\langle A \rangle\rangle E \square_c \varphi]_{\mathcal{M} \uparrow(s^*, t), q'}\}$ is indeed equal to $\inf_{t \in \Sigma_{\text{Agt} \setminus A}} \sup_{q' \in \tau_{\mathcal{M} \uparrow(s^*, t)}(q)} \inf_{t' \in \Sigma_{\text{Agt} \setminus A}} \{[\langle\langle A \rangle\rangle E \square_c \varphi]_{\mathcal{M} \uparrow(s^*, t'), q'}\}$. Thus, we obtain that A have at least as good options in R as in L , and hence $L \leq R$.

For the other direction, note that s, t in R are only relevant wrt the agents' actions in state q (later s', t' will be used). By unfolding R , we obtain an infinite sequence of collective action profiles $s_n(q_n), t_n(q_n)$ which maximize (over A 's actions) and minimize (over $\text{Agt} \setminus A$'s actions) the value of $\langle\langle A \rangle\rangle E \square_c \varphi$ in the next step. Now we observe that, when the system returns to state q , the same strategies s, t will be again optimal for the respective parties since the same expression will be maximized/minimized. Thus, the sequence of action profiles can be combined into a single pair of memoryless strategies s^*, t^* , which maximizes/minimizes $\langle\langle A \rangle\rangle E \square_c \varphi$ as good as the original sequence of strategies. In consequence, also $R \leq L$.

5 Conclusions

We extend the Markov Temporal Logic MTL from [11] to handle Markovian models with multiple agents acting in parallel. In terms of formal results, we show that the resulting logic strictly embeds ATL_{Ir^*} , i.e., alternating-time temporal logic with memoryless strategies. We also present fixpoint characterizations for some natural combinations of strategic, path, and temporal operators, that can be seen as analogues of Bellman equation. The characterizations enable computing the truth values of many MTL_2 formulae by solving sets of simple equations.

Acknowledgements. The research was partially conducted within the Polish development project no. O R000024 04.

References

1. Jamroga, W.: A temporal logic for Markov chains. In: Proceedings of AAMAS 2008, pp. 697–704 (2008)
2. Jamroga, W.: A temporal logic for multi-agent MDP's. In: Proceedings of Workshop on Formal Models and Methods for Multi-Robot Systems, pp. 29–34 (2008)
3. Zadeh, L.: Fuzzy sets. *Information and Control* 8(3), 338–353 (1965)
4. Nilsson, N.J.: Probabilistic logic. *Artificial Intelligence* 28(1), 71–87 (1986)
5. Ruspini, E., Lowrance, J., Strat, T.: Understanding evidential reasoning. *Artificial Intelligence* 6(3), 401–424 (1992)
6. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. *Formal Aspects of Computing* 6(5), 512–535 (1994)
7. Aziz, A., Singhal, V., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: It usually works: The temporal logic of stochastic systems. In: Wolper, P. (ed.) CAV 1995. LNCS, vol. 939. Springer, Heidelberg (1995)
8. Halpern, J.Y.: A logical approach to reasoning about uncertainty: a tutorial. In: Arrazola, X., Kortz, K., Pelletier, F.J. (eds.) *Discourse, Interaction, and Communication*, pp. 141–155. Kluwer, Dordrecht (1998)
9. Godefroid, P., Huth, M., Jagadeesan, R.: Abstraction-based model checking using modal transition systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 426–440. Springer, Heidelberg (2001)
10. Easterbrook, S., Chechik, M.: A framework for multi-valued reasoning over inconsistent viewpoints. In: International Conference on Software Engineering, pp. 411–420 (2001)
11. Konikowska, B., Penczek, W.: Model checking for multi-valued computation tree logic. In: Fitting, M., Orłowska, E. (eds.) *Beyond Two: Theory and Applications of Multiple Valued Logic*, pp. 193–210 (2003)
12. de Alfaro, L., Faella, M., Henzinger, T., Majumdar, R., Stoelinga, M.: Model checking discounted temporal properties. *Theoretical Computer Science* 345, 139–170 (2005)
13. Lluch-Lafuente, A., Montanari, U.: Quantitative μ -calculus and CTL based on constraint semirings. *Electr. Notes Theor. Comput. Sci.* 112, 37–59 (2005)
14. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) *Handbook of Theoretical Computer Science*, vol. B, pp. 995–1072. Elsevier Science Publishers, Amsterdam (1990)
15. Markov, A.: Rasprostranenie zakona bol'shikh chisel na velichiny, zavisyaschie drug ot druga. *Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete* 2(15), 135–156 (1906)
16. Kemeny, J.G., Snell, L.J., Knapp, A.W.: *Denumerable Markov Chains*. Van Nostrand (1966)
17. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time Temporal Logic. *Journal of the ACM* 49, 672–713 (2002)
18. Bellman, R.: A Markovian decision process. *Journal of Mathematics and Mechanics* 6, 679–684 (1957)
19. Bellman, R.: *Dynamic Programming*. Princeton University Press, Princeton (1957)
20. Boutilier, C.: Sequential optimality and coordination in multiagent systems. In: Proceedings of IJCAI, pp. 478–485 (1999)
21. Schobbens, P.Y.: Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science* 85(2) (2004)
22. Moller, F., Rabinovich, A.: On the expressive power of CTL*. In: Proceedings of LICS 1999, pp. 360–369 (1999)

Applying the Logic of Multiple-Valued Argumentation to Social Web: SNS and Wikipedia

Shusuke Kuribara¹, Safia Abbas¹, and Hajime Sawamura²

¹ Graduate School of Science and Technology, Niigata University
8050, 2-cho, Ikarashi, Nishi-ku, Niigata, Japan
{kuribara,safia}@cs.ie.niigata-u.ac.jp

² Institute of Science and Technology, Academic Assembly, Niigata University
8050, 2-cho, Ikarashi, Nishi-ku, Niigata, Japan
sawamura@ie.niigata-u.ac.jp

Abstract. The Logic of Multiple-Valued Argumentation (LMA) is an argumentation framework that allows for argument-based reasoning about uncertain issues under uncertain knowledge. In this paper, we describe its applications to Social Web: SNS and Wikipedia. They are said to be the most influential social Web applications to the present and future information society. For SNS, we present an agent that judges the registration approval for Mymixi in mixi in terms of LMA. For Wikipedia, we focus on the deletion problem of Wikipedia and present agents that argue about the issue on whether contributed articles should be deleted or not, analyzing arguments proposed for deletion in terms of LMA. These attempts reveal that LMA can deal with not only potential applications but also practical ones such as extensive and contemporary applications.

Keywords: Argumentation, Logic of Multiple-Valued Argumentation, Social Web, SNS, Wikipedia.

1 Introduction

Social software is reshaping the world we live in. In these days, much attention has been paid to social Web applications that link people, organizations, and concepts, instead of linking documents only. To cite a few, MySpace [1] and mixi [2] in SNS (Social Networking Service), Amazon [3] in information transmission service, Youtube [4] and Flickr [5] in information sharing service, Wikipedia [6] in user-originated information providing service.

In this paper, we first focus on SNS that will influence a new form of information society and a way of communication. Next we focus on Wikipedia that will be influential in forming free use of information and sharing mankind's common information.

For SNS, we will take up mixi that has the most SNS users in Japan. The mixi that provides members-only closed network can manage the friend list of a user by Mymixi that is a function of the mixi. The addition of a user to the

friend list by Mymixi is performed if the user makes a registration request to other users concerned by Mymixi and they approve it. The registration request might be sent by an unacquainted user who can be pernicious. So users are apt to hesitate to register him or her doubtlessly, even if they have a desire to communicate with many people equally. Thus a computer-supported method to advise us if the registration request should be accepted is desirable in such a dubious environment as faceless Internet.

On the other hand, for Wikipedia we will pay attention to the deletion problem for articles contributed to it. Although there are official policies and guidelines that all users who edit an article should follow, there appear users who do not follow them, and hence articles that should not be published in Wikipedia. To deal with such a problem, the so-called deletion policy is arranged in advance. When a user requests to delete articles that conform to the deletion policy by declaring "articles for deletion", the users who feel strongly about the issue enter into the discussion for deciding if pertinent articles should be deleted or not. Although there exists a record of argumentation for deletion in the form of articles, it is just a raw text in natural language, and hence it is a cumbersome or laborious task for us to analyze the arguments and yield a final status from those arguments. In Wikipedia, the final judgment seems to be made only by the administrator for the moment. We think the argument analysis and final decision should be made in a more open and objective manner.

In this paper, we address ourselves to solving the above problems by incorporating Logic of Multiple-valued Argumentation (LMA) [7] into SNS and Wikipedia. LMA is our own argumentation framework that allows for argument-based reasoning about uncertain issues under uncertain knowledge, and is successfully applied to the semantic Web reasoning [8] as well. Significances of argumentation have been recently recognized and logical models of it have emerged as a promising paradigm for modeling agent reasoning and communication in light of interaction with other agents and arguments [9][10][11]. Among other things, it has such a nature that closely mirrors the way humans reason, and hence provides us a general framework for inference and decision making in the presence of inconsistent, uncertain and incomplete information.

This paper is organized as follows. In Section 2, we outline the knowledge representation language, the Extended Annotated Logic Programming (EALP) for arguing agents, and the argumentation framework LMA. In Section 3, based on EALP and LMA, we describe the following three methods for agents: to decide if the registration requests should be approved or not in Mymixi, to decide if friends should be deleted or not from Mymixi, and to decide if additional proposals to Mymixi should be accepted or not. They could help us to live with a safe and trustworthy SNS, widening a circle of communication. In Section 4, we present an approach to the deletion problem in Wikipedia, based on EALP and LMA. Visualizing the process of an argument in a form of tree structure helps us analyze and understand it. Also for arguments on the deletion whose final statuses have remained undecided for a long time, our method is expected to

yield definite answers for them in an objective way. The final section summarizes the main contributions of this paper, and discusses some future works.

2 Knowledge Representation and Argumentation

In this section, we outline the EALP (Extended Annotated Logic Programming), and LMA. In Fig. 1, $\sim \text{positive}(W):0.7 \leftarrow \text{community_num}(W, X):1.0 \& (X < 20):1.0$ is a rule of EALP form. We read $\sim \text{positive}(W):0.7$ that (I wouldn't rather accept that W is positive to mixi). In EALP, we use two negation, \sim and **not**. \sim is the ontological explicit negation. **not** is a default negation.

Argumentation is a finite nonempty sequence of moves, $\text{move}_i = (\text{Player}_i, \text{Arg}_i)$ ($i \geq 1$) such that

1. $\text{Player}_i = P$ (Proponent) iff i is odd; and $\text{Player}_i = O$ (Opponent) $\Leftrightarrow i$ is even.
2. If $\text{Player}_i = \text{Player}_j = P$ ($i \neq j$) then $\text{Arg}_i \neq \text{Arg}_j$.
3. Arg_{i+1} defeats Arg_i .

In this paper, we define and use the following typical attack relations between two arguments. Justified arguments can be dialectically determined from a set of arguments by the dialectical proof theory. To facilitate understanding for justified arguments, we use dialogue tree. The dialogue tree is a tree of moves such that every branch is a dialogue. If termination of every branch of the dialogue tree with Arg as its root is a move of proponent, argument Arg is a provably justified argument.

1. Arg_1 rebuts $\text{Arg}_2 \Leftrightarrow$ there exists $A: \mu_1 \in \text{concl}(\text{Arg}_1)$ and $\sim A: \mu_2 \in \text{concl}(\text{Arg}_2)$ such that $\mu_1 \geq \mu_2$, or exists $\sim A: \mu_1 \in \text{concl}(\text{Arg}_1)$ and $A: \mu_2 \in \text{concl}(\text{Arg}_2)$ such that $\mu_1 \leq \mu_2$.
2. Arg_1 undercuts $\text{Arg}_2 \Leftrightarrow$ there exists $A: \mu_1 \in \text{concl}(\text{Arg}_1)$ and **not** $A: \mu_2 \in \text{assm}(\text{Arg}_2)$ such that $\mu_1 \geq \mu_2$, or exists $\sim A: \mu_1 \in \text{concl}(\text{Arg}_1)$ and **not** $\sim A: \mu_2 \in \text{assm}(\text{Arg}_2)$ such that $\mu_1 \leq \mu_2$.
3. Arg_1 defeats $\text{Arg}_2 \Leftrightarrow \text{Arg}_1$ undercuts Arg_2 , or Arg_1 rebuts Arg_2 and Arg_2 does not undercut Arg_1 .

3 Application to SNS

In this section, we describe three methods for maintaining the friend list of a user, Mymixi (from now on, we identify Mymixi with a friend list). Then, an agent's own evaluation or preference criterion and mixi users' profiles are to play a critical role in those three methods. An example of the evaluation criterion in terms of EALP is shown in Fig. 2 as a knowledge base, where the complete lattice of truth-values employed for the example is $\mathfrak{R}[0, 1]$, a unit interval of reals. Truth-values represent degree of belief for agents. In the profile specification of mixi, Fig. 2 compares one user's profile with the user agent's one, from which the agent can design an annotation for the literal $\text{correspond_hobby}(W)$, for example. The agent could assign to the literal an annotation of high degree if it corresponds with the specific hobby, or if they share many hobby with the agent. If birth place is the same in the both sides, it might sets 1.0 as an annotation to the literals $\text{same_born_place}(W)$. For the literal $\text{update_interval}(W)$ expressing

```

register(W):0.8 ← correspond_hobby(W):0.8&comment(W):0.7.
(If my hobby corresponds with W and W comments on my diary,
 I'll register W on My mixi.)
correspond_hobby(W):0.8 ← hobby(W, music):1.0.
(If W's hobby is listening to music, I correspond with W.)
comment(W):0.7 ← my_mixi(W, X):1.0&(X < 70):1.0.
(If the numbers of friends on W's Mymixi is less than seventy,
 I'll comment on me.)
register(W):0.6 ← same_birth_place(W):1.0.
(If W's birth place is the same as mine, I'll register him on W on Mymixi.)
~ comment(W):0.5 ← ~positive(W):0.7.
(If it's not true that W is positive to mixi arguably, W doesn't comment.)
~ positive(W):0.7 ← ~ community_num(W, X):1.0&(X < 20):1.0.
(If W's number of community is less than 20, W isn't positive.)
positive(W):0.7 ← update_interval(W):1.0.
(If W's frequency of the diary update is high, W is positive to mixi.)
    
```

Fig. 1. Knowledge base in EALP

name	age	gender	hometown	hobby	frequency of update	Mymixi	community
Kenji	38	male	tokyo	music	high	35	8
user	23	male	nigata	music	low	22	76

Fig. 2. Profile

```

KB1 = { delete(page_r) : t ←
        quotation_requirement(page_r) : f . }
(Delete) Insufficient quotation requirement
KB2 = { ~ delete(page_r) : t ←
        quotation_requirement(page_r) : t . }
(Continue) Sufficient quotation requirement
KB3 = { delete(page_r) : t ←
        quotation_requirement(page_r) : f . }
(Delete) No quotation written
KB4 = { ~ delete(page_r) : t ←
        abuse_of_delete_request(page_r) : t . }
(Continue) Abuse of delete request
    
```

Fig. 3. Four Agents' KB

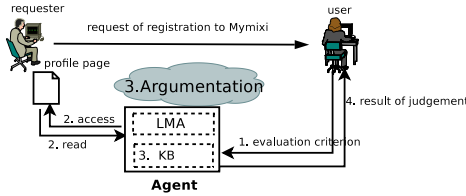


Fig. 4. Mymixi registration judgement framework

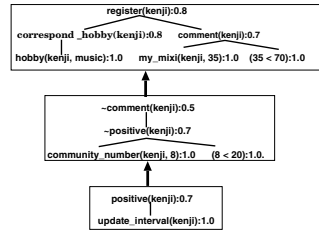


Fig. 5. Dialogue tree about Kenji

the update frequency of the diary in mixi, the agent figures the update frequency from the dates of the diary, and determines the annotation for the literal from it. For example, if a user (requester) updates the diary once in every month, the update frequency may be viewed high, and the annotation of $update_interval(W)$ would be 1.0. If a user (requester) updates every other month, the annotation of $update_interval(W)$ 0.5. In such a manner, associating annotations with literals was automated in our agent system for Mymixi management.

3.1 Examining the Registration Requests to Mymixi

We assume that the agent has the argument engine (LMA). Then, the registration requests to Mymixi are examined in the following steps. The overall Mymixi registration judgement framework is depicted in Fig. 4

- Step 1.** We prepare as a part of knowledge base an evaluation criterion \mathcal{E} that represents our preference for judgement in EALP, and leave it with our agent.
- Step 2.** When receiving a registration request from a requester, the agent accesses the profile page of the requester, and extracts information needful for the registration determination from it, and construct a set of rules of the form of EALP \mathcal{F} .
- Step 3.** The agent finally builds a knowledge base \mathcal{KB} that unites \mathcal{F} with the agent's evaluation criterion \mathcal{E} . With it, the agent starts argumentation on the issue whether to add the requester to Mymixi or not.

Step 4. The agent reports a result of judgement on the basis of the result of the argumentation to his/her master. The agent recommends to register the requester W on Mymixi to his/her master if the issue of the form $register(W) : \mu$ is justified. Otherwise, the agent does not recommend it.

Assume that we have received the registration request on our Mymixi from Kenji, and prepared a knowledge base \mathcal{KB} in Fig. 4 to which the agent's evaluation criterion and their profile information have been united. A result of the argumentation about Kenji based on the dialectical proof theory is shown in a dialogue tree in Fig. 5, where each argument is enclosed by a frame, and the defeat relation among arguments are drawn by arrows. The argument whose conclusion is $register(Kenji) : 0.8$ is justified since every leaf on the paths in the dialogue tree is the proponent's move. Consequently, the agent recommends to register Kenji on Mymixi.

3.2 Examining the Deletion of Friends from Mymixi

Everything changes over time. Our evaluation criterion and profile information are not exceptional. When the evaluation criterion and profile information of the agent have changed, the agent rebuild a new \mathcal{KB} . For instance, for the update frequency of the diary and the change of the affiliation number to communities, the agent should examine every month for keeping our decision-making trustworthy. Then, the agent starts argumentation using the revised \mathcal{KB} as done in Section 3.1. If an argument whose conclusion is $register(W) : \mu$ is not justified, the agent recommends to delete W from Mymixi. We assume that the agent adds such a new evaluation criterion as $\sim register(W) : 0.6 \leftarrow info(W, -, -, Age) : 1.0 \& (Age > 30) : 1.0$. (if W is older than 30, the agent doesn't register W on Mymixi.) to the evaluation criterion of Fig. 4. Reexamining Kenji with the new \mathcal{KB} , the dialogue tree about Kenji becomes one as in Fig. 3.3. The argument whose conclusion is $register(Kenji) : 0.8$ fails to be justified as the leaf on the right path of the dialogue tree is not a proponent' move. Therefore, the agent recommends to delete Kenji from Mymixi.

It is noted that we face an issue similar to belief revision since the deletion is considered as a revision of knowledge base. But what we have dealt with here amounts to its simplest form of belief revision in the sense that preexisting rules are not deleted from the knowledge base. Nevertheless, the previous decision has supplanted the new one, revealing a non-monotonic inference in terms of argumentation.

3.3 Approving Additional Proposal to Mymixi

The previous two subsections have been concerned with refining Mymixi in one way or another. In this subsection, we consider a framework to enhance or enrich Mymixi by exploring it more positively. The framework for proposals for adding relevant people to Mymixi is shown in Fig. 6. It starts by our indicating the agent so that it searches Mymixi and constructs a list for candidates who are entitled to be added to Mymixi on the basis of the upper limit of the number of

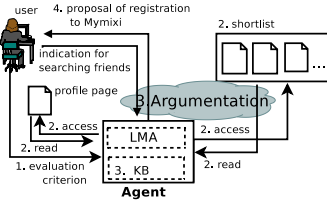


Fig. 6. Framework of additional proposal

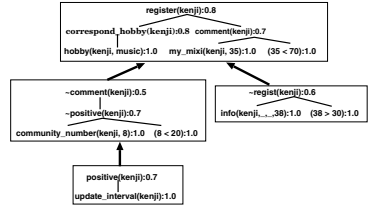


Fig. 7. Dialogue tree of reexamination about Kenji

people that we can accept. The agent examines it by the argumentation method that we provided in Section 3.1 for each candidate in the list. The agent recommends to add to Mymixi for such a member W of the list that the argument on $register(W) : \mu$ is justified. The search space is confined itself with Mymixi. But, introducing some mining techniques for arguments to this framework could lead to more fruitful search results [12].

4 Application to Wikipedia

In this section, we describe an application of LMA to Wikipedia. If a user requests to delete an article which conforms to the deletion policy of Wikipedia [6], any user can participate in the argument on whether the article should be deleted or not. The argument is done by voting on the deletion either of delete, continue or suspend together with the reason for it. Users also can put forward comments without voting. Here is a summary for them.

- (Delete) Users ballot for deletion, and present reasons why the article should be deleted.
- (Continue) Users ballot for continue, and present reasons why the article should be continued.
- (Suspend) Users ballot for suspension, and describes a reason why the user can not make a judgement of deletion nor continuation.
- (Comment) Users do not ballot for any alternatives above, but present comments only.

In this way, users’ opinions on the deletion are usually put forward, but sometimes counter-arguments to Delete and Continue are put forward in Comment as well, making it difficult for us to understand a logical structure of the argument at first sight. Administrators in Wikipedia are now entitled to make a final judgement on the matter whether to delete or not as a result of the argumentation, and in fact reach a decision by majority vote for Delete and Continue. But, there is no guarantee that the judgement has been held by a logical argumentation.

In order to support to analyze arguments on the deletion issues in a more logical and objective manner, we propose such a method that agents can resolve the deletion issues by argumentation in LMA with a record of the raw arguments

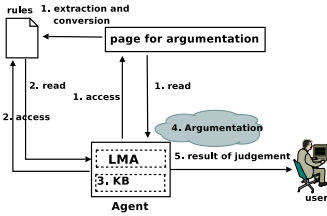


Fig. 8. Framework of deletion judgement

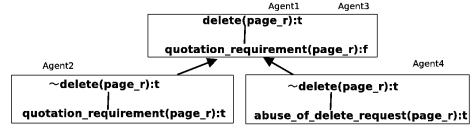


Fig. 9. Dialogue tree about "three laws of robotics"

converted to EALP. As the truth value for this application, we use the well-known complete lattice of the four values $FOUR = (\{t, f, \top, \perp\}, \leq), \forall x, y \in \{t, f, \top, \perp\} x \leq y \Leftrightarrow x = y \vee x = \perp \vee y = \top$. The t and f mean the alethic states of true and false respectively as in classical logic, and the \top and \perp mean contradictory and undecided states respectively. The alternative Suspend corresponds to the state/truth value \perp .

Fig. 8 illustrates a framework for deciding the deletion issues by the agents' argumentation. The agents have LMA as an argument engine as in the previous section. First of all, a user converts raw texts in the argumentation page for the deletion into EALP, and gives it to the agents. Then, assuming that the page P is requested for deletion, the agents start argumentation regarding the literal $delete(P) : \mu$ as an issue. If the argument on $delete(P) : t$ is justified, the agents judge that the page P should be deleted. Otherwise, they judge that it should be continued.

Let us take up a page on "three laws of robotics" which was requested for deletion in the past years and had been concluded that the page should be continued. First of all, the agents convert a text in the page into EALP. In the converted information (rules and arguments), the rules which especially relate to the argumentation is shown in table 3. We consider four agents $\{KB_1, \dots, KB_4\}$ since there were 4 people who substantially got involved in the argumentation. Fig. 9 displays the dialogue tree for judging "three laws of robotics" in which every leaves in it are not proponents' moves. From this, the agents turn out to such a judgement that "three laws of robotics" should be continued.

5 Conclusion and Future Work

In this paper, we proposed to apply the argumentation approach by means of the logic of multiple-valued argumentation (LMA) to two social Web applications: SNS and Wikipedia. Wherein, we have found the four approval issues in Mymixi and the deletion issue in Wikipedia most relevant to be resolved by argumentation from the nature of those issues, and provided the effective methods drawing on the power of argumentation for resolving them. We think that such reinforcement of SNS and Wikipedia with those augmentation is useful and helpful for users since they are now rapidly influencing our society as a new infrastructure for communication, information-sharing, decision-making, etc. in the next generation of information society.

There, however, are some important works left as future work. The agent in mixi was assumed to behave credulously in the sense that it believes user's profiles given are always true. It is usual that there exist users who are apt to fake their own profiles in such a dubious environment as faceless Internet. We need to take into account a somewhat careful or skeptical way to scrutinize profiles with the help of introductory essays on users, reputation in the community in which users participate, and so on. Work on trust and reputation are fortunately promoted in agent-oriented computing as well. On the other hand, a big bottleneck in the application of LMA to Wikipedia is that we assumed that the raw materials of arguments in natural language have to be represented in EALP by humans in this paper. We have had two preliminary approaches to overcome it. One is a promising work on transforming natural arguments in argument mapping systems to formal arguments in LMA [13]. The other is to construct and transform arguments by use of the argument mining from relational argument database [12]. Recent developments on Semantic Web technology [8] and Argument Interchange Format (AIF) [14] also turn out to be closely related to our present work, but the details of the tie-up will be left as a future work.

References

1. My Space! Japan, <http://jp.myspace.com/>
2. Mixi, <http://mixi.jp/>
3. Amazon! Japan, <http://www.amazon.co.jp/>
4. Youtube, <http://www.youtube.com/>
5. Flickr, <http://www.flickr.com/>
6. Wikipedia, <http://ja.wikipedia.org/wiki/>
7. Takahashi, T., Sawamura, H.: A logic of multiple-valued argumentation. In: Proceedings of the third international joint conference on Autonomous Agents and Multi Agent Systems (AAMAS 2004), pp. 800–807. ACM, New York (2004)
8. Sawamura, H., Wakaki, T., Nitta, K.: The logic of multiple-valued argumentation and its applications to web technology. In: Dunne, P.E., Bench-Capon, T.J.M. (eds.) Computational Models of Argument, pp. 291–296. IOS Press, Amsterdam (2006)
9. Chesñevar, C.I., Maguitman, G., Loui, R.P.: Logical models of argument. *ACM Computing Surveys* 32, 337–383 (2000)
10. Prakken, H., Vreeswijk, G.: Logical systems for defeasible argumentation. In: Gabbay, D., Guenther, F. (eds.) *Handbook of Philosophical Logic*, pp. 219–318. Kluwer, Dordrecht (2002)
11. Reed, C., Norman, T.J. (eds.): *Argumentation Machines*. Kluwer Academic Publishers, Dordrecht (2004)
12. Abbas, S., Sawamura, H.: Towards argument mining from relational argument database. In: Proc. of the 2nd Int. Workshop on Juris-Informatics (JURISIN 2008) Workshop, pp. 22–31 (2008)
13. Takahashi, Y., Sawamura, H., Zhang, J.: Transforming natural arguments in araucaria to formal arguments in lma. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 668–678. IEEE Computer Society, Los Alamitos (2006)
14. Chesñevar, C.I., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G., South, M., Vreeswijk, G., Willmott, S.: Towards an argument interchange format. *The Knowledge Engineering Review* 21, 293–316 (2006)

Simulation of Halal Food Supply Chain with Certification System: A Multi-Agent System Approach

YiHua Lam and Saadat M. Alhashmi

School of Information Technology, Sunway Campus, Monash University,
Bandar Sunway, 46150, Petaling Jaya, Malaysia
{Lam.yek.Wah, Saadat.M.Alhashmi}@infotech.monash.edu.my

Abstract. Certification of Halal food product supply becomes a challenging task as various validating procedures have to be undergone under the emergence of vast business networks in food supply chain. In order to maintain high quality assurance of every validating procedure, and to fulfill demand from immense religious population, highly efficient method will be needed to monitor, to record and to register, to decide and to certify every actor (agent) and every product in the supply chain. With the multi-agent architecture this research work simulates the Halal food supply chain planning with certification system, which attempts to replicate the actual market place coupled with Halal food quality requirement. Statistical study of the decision making of various agents in the supply chain and the response of certification system will verify the feasibility of the certification framework in supply chain.

Keywords: Multi-agent, Halal food, supply chain, certification process.

1 Introduction

Post-modern view of food quality is emerging from the factors of culture, environmental and ethical values, biological values, sensual and nutritional values, functional values and authenticity [1, 2, 3]. However, the understanding of specifically imposed values in food quality is not new in Islamic culture. Food permitted by Islamic dietary law is called as Halal food (permitted food). Products certified as Halal are products that not only abide by the religion aspect but include hygiene, sanitation and safety qualities [4] which are important quality aspects of food. The consumers of Halal products include Muslims and non-Muslims that hail from many countries such as Asian countries, Middle East countries, America, Canada, United Kingdom, Africa and Europe. As quality is associated with the Halal standard, it is of utmost importance that the products and meat sold are genuinely Halal, but many times it has been found that this is not the case in many countries [5, 6, 7, 8, 9, 10, 11, 12]. For instance, manufacturers and distribution outlets such as food sellers are using fake Halal certificates or labels on products and meat [8, 10, 11] there is cross contamination in the production of Halal and non-Halal food at the manufacturers [7, 9, 13], animals are not slaughtered ritually [7, 13], there is clumsy slaughtering of animals with dull knives [12] which is not only cruel but is considered not Halal under the MS1500:2004 standard [11] which adheres to the Shariah law (Islamic law) and has

been accepted by the United Nations, and animals are starved before the slaughtering according to slaughterhouse officials [14] as well as cruelly transported by traders such as from the supplier to the abattoir [12, 14] which is cruel and therefore not Halal as quoted by the late B.A. Hafiz al-Masri, Imam of the Shah Jehan Mosque in the United Kingdom [12] who has contributed that 'animals subjected to cruelties in their breeding, transport, slaughter, or in general welfare, meat from them is considered impure', in other words not Halal. This has caused the consumers of Halal products especially the Muslim consumers to be concerned about the quality of the Halal food and products that they consume and use on a daily basis.

Today authorized Muslim certification organizations have been set up in many countries to monitor and to inspect the abattoirs, manufacturers and distribution outlets in their handling of the animals and Halal products, and to issue the Halal certificates to them once the inspections are approved. These certification organizations have been successful to a certain extent in reducing the quality problem associated with the Halal products, but there is room for improvement. Currently, these organizations are carrying out their inspections manually and there are certification organizations that require companies to manually submit the Halal application to their offices and liaise with third party organizations/labs to test the Halal products, which delay the Halal certification process and ultimately delay the genuine Halal products from reaching the consumers' hands quickly. Procedural delays can cause all sorts of issues not just with consumers but along the supply chain as well. This paper proposes a multi-agent based approach to improve the Halal supply chain in general and the certification process in particular.

With the advancement of Multi-Agent System (MAS), we can verify and study the proposed *Halal* certification system framework by constructing a multi-agent environment on Repast [15] platform to replicate the real market space coexisting with authorized Muslim certification organization. In addition, this paper statistically presents the feasibility of a tested *Halal* certification framework to counter the procedural hindrances in certifying *Halal* products in food supply chain. A "*pseudo-synchronous*" method, which manipulates the shuffling technique, is implemented to imitate the concurrent decision-making process of every participating agent. This decision-making could be in the form of information retrieval, finding the best deal from various offered bids, procuring livestock supply and registering *Halal* food data. In our view, a multi autonomous agent environment must possess the autonomy and intelligence in decision-making. Hence, based on previous occurrences, current situation and its own interest, every agent has control over its internal state and actions. An autonomous agent, which possesses intelligent behavior, will execute the assigned tasks on its own for the achievement of predefined goals. Therefore, actions will be different among agents, which may be either the same type agent or different type agent. This individual decision maker will complicate the simulated food supply chain with *Halal* certification system. Moreover, technology can be great enabler especially when it comes to routine food quality inspection. By considering such advancement this paper puts forward a multi-agent based simulated framework for testing the *Halal* certification system in food supply chain.

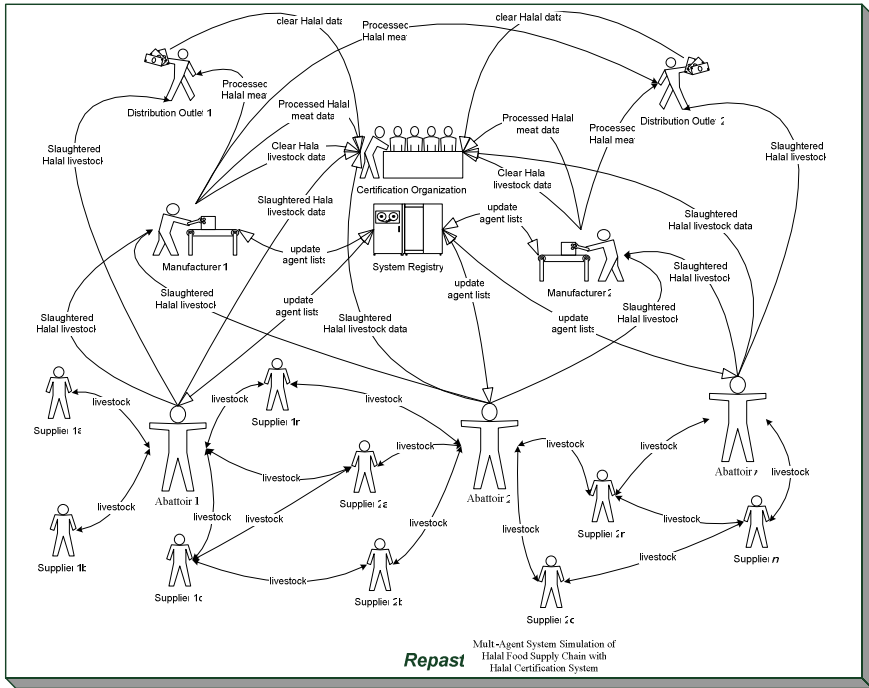


Fig. 1. Multi-agent environment overview

2 The Simulated Halal Food Supply with Certification System Model Space

Our focus is on replicating the basic real world problem by mapping it onto MAS simulation. The simulated Halal food supply chain with Halal certification system will demonstrate (1) complex interaction of various agents in Halal food supply chain, the reactions of agents with Certification Organization Agent (Cert. Org. Agent) and food business network; (2) feasibility of the certification information flow (3) statistical studies of certification efficiency; and (4) pseudo-synchronous interactions among multiple actors that require independent decision-making capability. Every agent is capable to "concurrently" make decisions at run time. Thus, the simulated Halal food supply with certification system will have to be pseudo-synchronous by using shuffling method with randomly selected sequence. By using shuffling technique, the sequence of interaction, Supplier/Breeder Agent→Abattoir Agent→Manufacturer Agent→Distribution Outlet can be shuffled according to random number. That means the sequence can be led by either a Supplier/Breeder Agent, or an Abattoir Agent, or a Manufacturer Agent, or a Distribution Outlet in every time step of Repast. However, Cert. Org. Agent cannot lead any sequence.

Every agent is able to perform independent actions on behalf of its own, and work out how to achieve its design objectives, autonomously and dynamically, because every agent is dealing with current situation and individual data. Every agent in the Halal supply chain system replicates basic functions of the real world. The different actors that have been identified in this framework are categorized in two groups, i.e. the group of Supply Chain Organizations, which includes Supplier Agent or Breeder Agent, Abattoir Agent, Manufacturer Agent and Distribution Outlets; and the group of Cert. Org. Agent (and System Registry).

Every agent will autonomously perform in this system. For instance, the Abattoir Agent replicates the basic action of butcher. It will procure livestock from Supplier/Breeder Agents, who supply/breed the livestock, select, slaughter and sell them to Manufacturer Agents or Distribution Outlets. Each Representative (interface) of Cert. Org. Agent residing in Abattoir Agents will select/justify Halal and non-Halal livestock before they will be slaughtered by Abattoir Agent. The selected and slaughtered livestock will be labeled as Halal by the Representative as well. Once the slaughtered Halal livestock is sent to buyer, Abattoir Agent will submit slaughtered Halal livestock data to Cert. Org. Agent. An Abattoir Agent may sell slaughtered Halal livestock to a Manufacturer Agent or a Distribution Outlet. If a Manufacturer Agent purchases the slaughtered Halal livestock from an Abattoir Agent, it processes and butchers the slaughtered Halal livestock into pieces of meat. Again, a Representative of Cert. Org. Agent residing in that Manufacturer Agent will label every processed meat before it will be sold to a Distribution Outlet which may be a Food Retailer, or a Butcher or a Hypermarket. Information of processed Halal meat from Manufacturer Agent will be sent to Cert. Org. Agent as well via a Representative. As basically, there are three types of Distribution Outlet, hence different type of Distribution Outlet will have different rate of selling Halal food. For instance, Food Retailer will have higher rate to sell processed Halal meat compared to Butcher, whereas Butcher will have higher rate to sell slaughtered Halal livestock compared to Food Retailer. However, Hypermarket will have an almost balanced Halal food selling rate.

The flow of financial point (money) is from Distribution Outlet to Supplier Agent via Manufacturer Agent and Abattoir Agent. That means Distribution Outlet will sell Halal food and generate money. When a Distribution Outlet purchases processed Halal meat from a Manufacturer Agent or buys slaughtered Halal livestock from an Abattoir Agent, it will pass money to either one of them. Money will flow from Manufacturer Agent to Abattoir Agent when trading occurs. Finally, Supplier Agent will accumulate money from Abattoir Agents and use money to generate livestock. Contrary, the flow of livestock is from Supplier Agent to Distribution Outlet. Furthermore, two types of Halal food data, i.e. slaughtered Halal livestock and processed Halal meat, will be kept in Cert. Org. Agent database. The slaughtered Halal livestock data kept in Cert. Org. Agent will only be cleared if the slaughtered Halal livestock has been processed by Manufacturer Agents or it has been sold at Distribution Outlets. The processed Halal meat data will only be cleared from Cert. Org. Agent database if it has been sold at Distribution Outlet.

In addition, the simulated livestock market involves Supplier/Breeder Agents and Abattoir Agents. That means, after breeding livestock, the Supplier Agent not only trade livestock with Abattoir Agent, but it may also trade with other Supplier Agents as well. The higher the offered bid given by an agent, the higher the probability an

agent will get the livestock. The same business network will happen when an Abattoir Agent opens its sale to Manufacturer Agents and Distribution Outlets. Moreover, every agent will tend to sustain its lifespan by continuously purchasing and selling to generate more money. It will become idle when it has low amount of money, and it will be erased from the Halal food supply chain model space when its life is expired. Agents will tend to lift up the offer to outbid other agents in order to succeed in food market. The objective of constructing such “trading” is to complicate the food supply business network. Therefore, every agent will be lively to trade around in the Halal food market. Furthermore, once an agent is added to the Halal food supply chain, it will receive a certain period of permit (e.g. 50 time step) from Cert. Org. Agent. Random examination of Halal food will be carried out by Cert. Org. Agent.

3 Results

The model space of Halal food supply chain with Certification System Agent is depicted in Fig. 2. Every symbol is labeled and numbered accordingly with the type of agent and an index number.

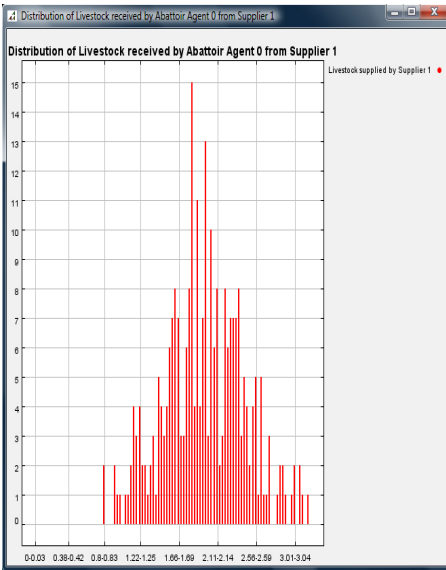


Fig. 2. Distribution of livestock received by Abattoir Agent 0 from Supplier Agent 1

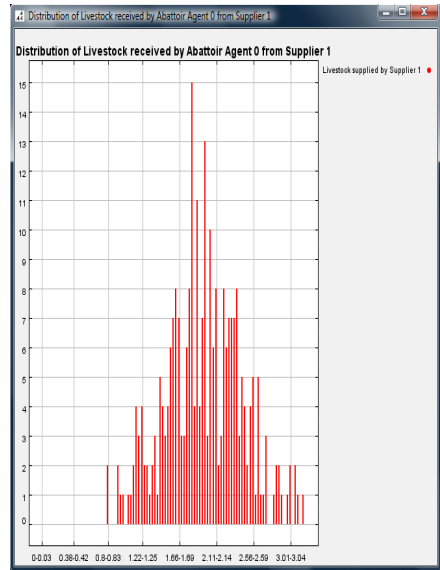


Fig. 3. Distribution of Halal livestock received by Abattoir Agent 0 from Supplier Agent 1

Fig. 2 shows interactions happen among the same type agents (Supplier Agent interacts with Supplier Agent) and different type agents (e.g. Abattoir Agent interacts with Manufacturer Agent or Distribution Outlet and Cert. Org. Agent) within one time step. Such interactions occur randomly and strategically under autonomous decision-making process of agents with probabilistic consideration as described in

section 2. Red links represent trading interaction; whereas blue and cyan links depict passing/sending of slaughtered Halal livestock information and processed Halal meat information to Cert. Org. Agent; in addition, orange links portray clearings of Halal food data at Certification Organization database which are requested by Distribution Outlets. Besides, Fig. 2 shows various types of agent lively participate in the Halal food supply business network within one time step. This complication of network replicates the basic actual Halal food supply chain Fig. 3 shows a livestock distribution delivered by Supplier Agent to Abattoir Agent in one time step. In this paper, weight of livestock is assumed and used as the deciding factor to justify whether the livestock is Halal or non Halal. The distribution is recorded by Abattoir Agent once it has received livestock supply from Supplier Agent. After receiving livestock, Abattoir Agent assumes the Halal requirement of livestock weight is above or is the same as 1.5 weight unit. Obviously, Fig. 3 shows the livestock recorded by Abattoir Agent is in accord with Gaussian distribution and the distribution reproduces what has been generated by Supplier Agent with standard deviation 0.5, mean value 2.0 Fig. 4 significantly shows the selected livestock distribution at Abattoir Agent is above 1.5 weight unit.

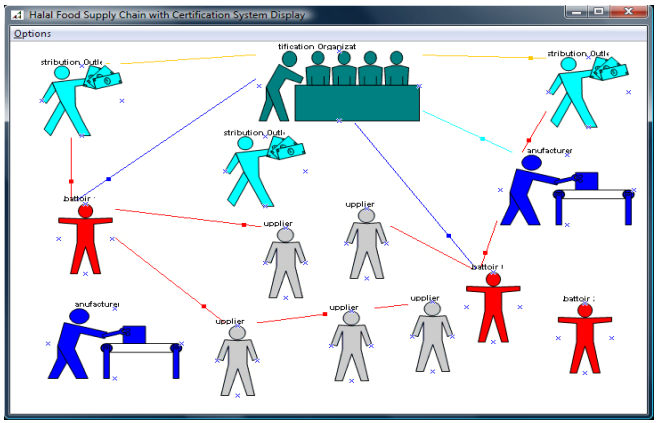


Fig. 4. Halal food supply chain with Certification System Agent

4 Conclusion and Discussion

As stated in the introduction, the vast religious population of Islamic culture, who impose high requirement on food, demands the need to have a high efficiency and fast network base food certification system. However, before the real implementation of network base food certification system, Halal food supply chain planning with Certification System can be tested in a MAS environment. For instance, the color links of transferring Halal data in Fig. 2 may represent connections of e-business (ebXML) [16, 17] or RosettaNet [18]. The system, from a system-design perspective, shows evidence of a flexible design. What this means is that it possesses the elements of scaling up, functioning in a distributed environment, and above all, autonomy. This puts forward a framework where multi-agents can replicate the basic actual environment

and test the feasibility of a certification system on top of a food supply chain. The MAS model space has tested with five types of agent. Definitely, a larger number of agents either the same type or different type or combination of various types (e.g. contemporary hypermarket which consists of food retailer and abattoir) will probably increase the similarity with actual environment and reliability of the simulation, meanwhile augment the complexity of the virtual food business network. On the other hand, having a fewer number of agents will probably not bring out the advantages of a multi-agent approach, and the system could end up not achieving the main purpose as it was intended to. The focus of this MAS system development was on designing a generic and robust system. In the real world, the actual process of implementation depends on external variables such as market trend, behavioral, ethical, economical and social factors. However, this MAS system only considered the technical aspects, and at least this would pave the way by providing a framework where the above mentioned variables can be considered and explored for testing the real world application. This research will be undertaken to improve decision-making in certifying food supply under behavioral economic consideration and consumer trend. Evaluation in the real world also depends on other external factors such as: legislation, economic constraints, social and ethical aspects, for example. Nevertheless, the developed system will provide a framework in which these issues and problems of Halal food supply chain with certification system can be identified and articulated more easily.

References

1. Eastham, J.F., Sharples, L., Ball, S.D.: Food Supply Chain Management: Issues for the Hospitality and Retail Sectors, Butterworth-Heinemann, p. 22 (2001)
2. Woodward, L., Stolton, S., Dudley, N. (eds.): Food Quality: Concepts and Methodologies. In: Proc. Colloquium Elm Farm Research Center, Newbury, UK (1989)
3. Woodward, L., Meier-Ploeger, A.: Consumer Perceptions of Organic Food Quality. In: IFOAM Conf., Mal Del Plata, Argentina (1998)
4. SIRIM, http://www.sirim.my/f_corp/july04.pdf
5. American Communications Foundation,
<http://www.acfnewsresource.org/religion/Halal.html>
6. Halal Journal, http://www.Halaljournal.com/artman/publish_php/article_1036.php
7. Halal Monitoring Authority,
http://hma.jucanada.org/industry_problems.aspx
8. Halal Monitoring Committee,
http://www.Halalmc.co.uk/about_hmc/need_monitoring.html
9. Islamic Food and Nutrition Council of America,
http://ifanca.org/newsletter/2002_05.htm
10. Islamic Religious Council Singapore, <http://www.muis.gov.sg/webpub/warita/warita3-2000/page22.html>
11. JAKIM, <http://www.islam.gov.my/portal/lihat.php?jakim=2140>
12. PETA, <http://www.petaindia.com/600murator.html>
13. Halal Monitoring Committee,
http://www.Halalmc.co.uk/about_hmc/industry_problems.html

14. People of the Ethical Treatment of Animals, <http://www.petaindia.com/dreport2.html>
15. North, M.J., Collier, N.T., Vos, J.R.: Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit. *ACM Transactions on Modeling and Computer Simulation* 16(1), 1–25 (2006)
16. ebXML (2001a), <http://www.ebxml.org/specs/ebTA.pdf>
17. ebXML (2001b), http://www.ebxml.org/specdrafts/ebbpss_v1.0.pdf
18. Rosetta Net, <http://www.rosettanet.org>

A Novel Approach for Conflict Resolution in Context-Awareness Using Semantic Unification of Multi-Cognition^{*}

Keonsoo Lee¹ and Minkoo Kim²

¹ Graduate School of Information and Communication, Ajou University,
Suwon, Kyonggido 442-749, Republic of Korea
lks7256@ajou.ac.kr

² College of Information & Computer Engineering, Ajou University,
Suwon, Kyonggido 442-749, Republic of Korea
minkoo@ajou.ac.kr

Abstract. The context helps service providers to perform their works properly even if the user's request has implicit meanings. The context of service can be viewed as the situation of the environment which is related to the service's execution. In order to provide such autonomous and intelligent service, the provider needs to be aware of the context by sensing the environment. In this sensing process, the conflicts can happen. One of the origins of these conflicts is the discord of the sensing data's meaning that is used for recognizing the context by different providers. This difference can cause the conflict in providing services such as keeping the music or keeping the serenity of the space. In this paper, we propose a conflict resolution method by unifying these semantic discards of sensed data. This unified meaning can keep the multiple services from recognizing the environment differently.

1 Introduction

In origin, the ubiquitous computing promises the limitless access to any networks. But these days, more than this feature are expected for this novel computing environment. More autonomous and more intelligent service is the expected one. This service needs to perform according to the dynamically changed environment. This intelligent response to the environment means that the service knows when to ignite itself and when to stop. At the same time, the same service can execute differently according to the situation where it performed. In ubiquitous computing environment, this service works according to the dynamically changed situation of restaurant without the customers or manager's explicit request. This performance can be called autonomous and intelligent [1]. As the computing system gets to be more complex, it gets to be more difficult for user to operate the system. With this amateurism of the users for the

^{*} This research is supported by Foundation of ubiquitous computing and networking (UCN) Project, the Ministry of Knowledge Economy (MKE) 21st Century Frontier R&D Program in Korea and a result of subproject UCN 08B3-S2-10M.

computational system, the system should have knowledge for understanding the user's ambiguous command. A context works as the knowledge for the service's intelligent performance. By using the context, the execution of a specific service can be controlled delicately to satisfy the user even if the request of that user has some omitted details, implicitly hidden desires, and ambiguousness. The context can be used as one of the most important foundations in the determination of services and their executing conditions.

One of the most important admonishments in using a context is the possibility of conflicts. In ubiquitous computing environment, where various users exist and their desires are entangled with others, the mutually exclusive services can be fired by the same context. The temperature of 25°C can be a factor of the context for not only the service of using an air-conditioner but also the service of using a heater. This disagreement for the meaning of context results the conflict and it prevents the service from satisfying the requesters. In order to resolve these conflicts, we propose a semantic unification method that coordinates the meaning of the context recognized by different services.

2 Background

2.1 Context-Awareness

Becoming aware of the context can cause some advantages not only in ubiquitous computing environment but also in other domains. In the lexical meaning, contexts help to extract the correct meaning of the concept from the paragraph. In this case, the tone, pitch, volume, and face can tell the intended meaning. The same situation happens in computing areas especially ubiquitous computing where the various services and multiple users exist. In such case, the context information can be usefully employed to provide the proper service by eliminating the ambiguity of services' execution condition (pre-condition). The general system model that responds to the environment employs these process. A service provider senses the environment. From the sensed information, he/she can recognize the situation and decide whether the service is proper to the situation or not. If the situation matches the execution condition of the service, the service is viewed as proper service and it is executed. In this process, the context works as yardsticks for matching the situation and service's execution condition.

2.2 Context, Service and Conflict

Every service needs to use contexts for suitable execution in given environment. The information for a specific service's context is different from that of other services. As the Fig 1 shows, the physical environment is viewed as a model consists of the existing computational objects. The snapshot of the model at a specific time t is called the situation of the environment at time t . As the time passes, the situation is changed and the model represents the environment [2]. In this situation model, the context can be seen as the subset of situation. As mentioned before, for a temperature control service, the brightness sensing information is not necessary. Therefore, the context of a service is a part of the situation which consists of the states of computing objects that the service concerns.

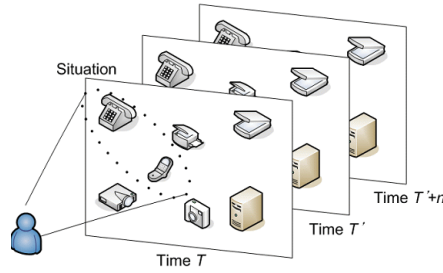


Fig. 1. Contexts from Situation

A service is an activity with an identified goal. The goal is a desired state of the current environment. Therefore, we can define the service as a set of actions for changing the current environment to the desired state. In order to use context for providing services, the service provider should know which information is necessary. This information is charged by the service provider. When new service is registered, its context frame should be included even if the necessary information is not able to be gathered in given situation. Fig 2 shows the description frame of a service definition. It consists of execution condition, context information (set of necessary sensing targets), post-condition (the desired change to the situation), and action flow.

Execution Condition	Context Frame	Post Condition	Action Flow
The condition of when the service can be provided and executed	State of Object 1. Object 1 2. Object 2 ...	Change of Situation resulted by this service execution	Necessary Device and its usage sequence

Fig. 2. Description frame of a service definition

In the process of providing services with their proper contexts, conflicts can occur. The conflict is a situation where more than two services, which cannot coexist, are tried to execute at the same time. In order to execute several services parallel, each service’s goal state and employed device should not be overlapped. Let me assume two services. One service tries to show a movie. The other service tries to show a soap opera. These two services’ goal state is same. Both of them want to change audio of the environment more entertaining. However, when there is only one device such as television, which is employed for archiving the desired services, the conflict occurs. The dependency between service and the environment, between service and its employing devices causes the conflict.

In order to resolve such conflicts, there are two different approaches. One is the resolution at the service execution time. The other is the prevention at the service selection time [3]. Most widely employed resolution method is using priority at the service execution time. When a conflict occurs, more important service, which has a higher priority, performs ahead. The determination of each service’s priory is the key point of relevant result. In this method, new plan is generated for every new request.

Then the consistency of conflict-free execution with other services is checked. After that, the service is carried out with the plan. This planning process for conflict-free service providing is called whenever the new request occurs. When the number of requests is numerous, that can be a burden to the system.

The other method checks the consistency checking before the plan for the service is made. As this method checks the conflict ahead, the load of the system can be deducted. The proposed method in this paper is in this category.

3 Proposed Method

When the execution condition of a service is matched to the current situation, the service is selected. This selected service is adjusted according to its context information. For example, let me assume a light control service. This service is lighting up or dimming down the area according to the current situation. When there is no one in that area, it turns off the light. When the TV is on, it dims down. When a user enters this area, it lights up. For this service, the execution condition is the gap between the current brightness and the necessary brightness. The necessary brightness is determined by the context of this service such as the existence of user, the operation of any devices that affects the brightness of this area like TV, beam projector, or computer monitor. The post condition will be the elimination of the gap between the current brightness and the intended one. The conflict about this service will occur when one user wants to turn off the light for sleeping and another user wants to turn of the light for reading. This means that the intended brightness of the first user is different from that of the second user. More generally, when the several services, which have mutually exclusive relation by their post condition, are selected at the same time, the conflict occurs.

The mutually exclusive relation among services can be known by their post condition factor. However, the origin of how these services can selected at the same time is the context frame information they use. As shown in the Fig 3, when the services' context frames are overlapped, the conflict may occur. The requester1 tries to select a service which uses the states of device1 and device2 as its context. The requester2 tries to select another service which uses the states of device 3, device4, and device 2 as its context. The state of device2 is recognized by different services. If request2 knows that the device2's state is used for request1's context with a specific meaning, request2 will change the service plan. Let me assume another example. There is a temperature service. The current temperature is 25°C. One user wants to use this service to heat up the area. The other user wants to use this service to cool down the area. This conflict can be resolved at service execution time calculating whose request is more important. However, if they know the meaning of the current temperature such as temperature 25°C means a warm state, the heating service will not selected in the first place. If the context data has a semantic meaning and this meaning is jointly recognized by services, the conflict can be resolved at service selection time not service execution time.

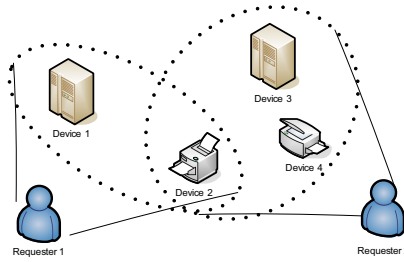


Fig. 3. Multi-cognition of the same object for different service's context

3.1 Semantic Unification

With a global view of the situation, the conflict resolution will be easier. However, it is a heavy burden for service providers to maintain the global view [4]. Therefore a novel approach for conflict resolution with local view is necessary. This semantic unification method is a way of sharing local information. For example, the heater can work at temperature 25°C. At the same time the air-conditioner can work at the same temperature. However, when it is cool, the air-conditioner will not operate. The proposed method gives a raw data a meaning. With this meaning, the service can be checked whether the conflict may occur before it is selected.

$P(M_i) = \frac{\sum_{k=1}^M P(U_k)}{N}$	M_i : i th meaning U_k : k th User who wants the M_i $P()$: Priority function N : Total number of users M : Total number of users who wants the M_i
--	--

Fig. 4. Priority Formula

Once the conflict is detected according to the meaning, the service is modified. For example, temperature 25°C has the meaning of warm, the heating service will not be selected. However, the air-conditioner which tries to cool the area may be selected. But the meaning warm is assigned by previous service and need to be remained warm. In such case, the cooling service can be performed without conflicts until the warm state is not changed. If the warm means the temperature between 22°C and 26°C, the cooling service can be performed until the temperature becomes 22°C without changing the meaning. Therefore how to mapping the raw data to the semantic terminology is one of the most important things in this method. Another feature needed to be considered is the priority of semantic assignment. When one service tries to remain a warm state and another service tries to change the state into the cool state, the priority of the request can be the answer. In order to compute the priority of the desired meaning, we propose this formula shown in Fig 4.

The flow of semantic unification follows these steps. First, multiply recognized context factors are retrieved. These factors can be found by scanning each service's context frame. Second, the service provider checks whether the assigned meaning of

the context factor exists when a service is selected. If the meaning is null, it assigns the service’s post condition meaning which is the desired state of the service. If the meaning is already assigned, the service provider calculates the priority of each meaning and changes it according to the calculating result. Third, if the service is against the meaning, the service is modified to archive the requester’s desire without changing the meaning or rejected. By using this semantic meaning of context factor, the conflict can be detected before the service is selected and executed.

3.2 Service Model with the Proposed Method

The proposed model is shown in Fig 5. This model consists of 4 factors; sensors, service list, sectioned blackboard and agents. The sensors are sensing the environment and make the situation. The service list is a kind of database of services which are possible in that environment. When a new service is registered, it stored in this list with the service format information shown in Fig 2.

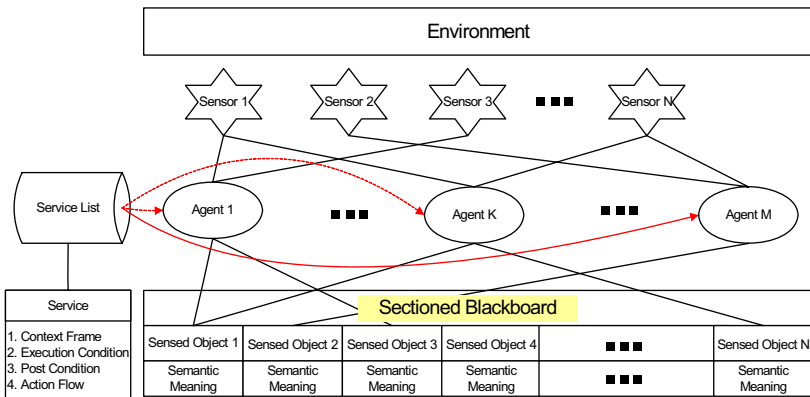


Fig. 5. Proposed Model

The sectioned blackboard is where the semantically unified information is saved. An agent is an essential member of this model which works as a service provider. Every agent has its own service and provides its service to any user who needs that service. When a new service is registered, it stored in service list and its agent is deployed. This agent connects to the sensors for the service’s execution condition and context. Once the sensed information is matched to the service’s execution condition, the agent checks the sectioned blackboard for retrieving its context meaning. As the blackboard is sectioned for the sensed data, its retrieving response time will be relevantly short.

If a service’s post condition does not affect the current meaning of context used by other service, it is selected and performed. If not, the service is suspended and modified. If the modification cannot satisfy the requester’s desire without changing the existing meaning, it will be rejected. With this semantic information the conflict can be detected before the service is selected.

4 Simulation and Results

In order to evaluate this proposed method, we assumed a simulation scenario and tested the proposed method in this scenario. Table 1 shows the simple simulated environment. In this environment, the temperature and brightness controls are served. In this environment, two users exist with same user priority.

Table 1. Simulated Environment

Environment				
Sensors	Temperature			
	Brightness			
Services	Service Name	Execution Condition	Context Frame	Post Condition
	Temperature Control	Temperature Gap	Air-Condition, Heater, Temperature, User's Location	Desired Temperature
	Brightness Control	Brightness Gap	Bed Lamp, Light, Brightness, User's Location	Desired Brightness
Devices	Air-Conditioner			
	Heater			
	Bed Lamp			
	Light			

The conflicts occur following the users' movement in this given environment. Firstly, this environment's situation is 30Lx and 18°C. When a user enters this area, the existence of the user changes the current situation. This change requests two services; temperature and brightness control. The brightness control service lights up the area and marks the semantic meaning of brightness as *Bright*. The temperature control service warms the area to 22°C and marks the semantic meaning of temperature as *Warm*. As the user sleeps, the brightness control service is operated and the semantic of brightness is changed to *Dark*. Then the second user enters this area. His/her existence tries to select the brightness control service. This service will produce the conflict to the first user's brightness control service. However, the semantic meaning of brightness factor marked as *Dark* prevents the second user's service from starting. From this semantic meaning, the service provider knows that this area should be remained dark. Therefore, the provider changes the service operation specification by using the bed lamp not light. Even if the lamp turned on, the brightness of this area is still *Dark*. Then the second user feels warm and the temperature control service is selected. As the previous brightness control service, this request is suspended and modified. If the modification can archive the second user's desire without changing the semantic meaning of temperature, the service is selected and performed.

The main factor of proper resolution is the way of mapping the raw data to the semantic meaning. If the *Dark* means bright state between 0Lx and 80Lx, and the bed lamp lights up the brightness over 80Lx, the second user's service will not be started. The same thing happens to the temperature control service. The goal of this proposed method is to notify the operation of other services to the newly generated service. How to determine the meaning of the operation is the system manager's responsibility.

5 Conclusion

The context-awareness is necessary for providing autonomous and intelligent service according to the dynamically changing environment. Nevertheless, the usage of contexts can result conflicts among services and this concludes the user's inconvenience and distrust to the system. In order to resolve these conflicts, several approaches are studied. The most effective approach is preventing any services, which may conflict with existing ones, from executing. As a service is selected, when the situation gratifies the execution condition of the service, we assign a semantic meaning to the sensed information that the service uses as its context. Once a service changes this meaning, the other services which use the same context can be aware not only the physical data but also the how it used by other services. With this semantic meaning, any services, that use the same context, can be aware of how other related services work and adjust their operation preventing conflict with other services.

The conflicts resulted from the parallel operation of multiple services which have mutually exclusive relation can be resolved by using this semantic unification of context.

References

- 1 Dey, A.K.: Providing Architectural Support for Building Context-Aware Applications. PhD thesis, College of Computing, Georgia Institute of Technology (2000)
- 2 Reiter, R.: The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, 359–380 (1991)
- 3 Lee, K., Kim, M.: Conflict Resolution Method for Multi-Context Situation. In: *The Eighth Pacific Rim International Workshop on Multi-Agents, Malaysia*, pp. 285–294 (2005)
- 4 Lee, K., Kim, W., Kim, M.: Resource Allocation in Multi-Agent System for Ubiquitous Computing Service. In: *The Eighth Pacific Rim International Workshop on Multi-Agents, Malaysia*, pp. 113–120 (2005)
- 5 Levesque, H., Pirri, F., Reiter, R.: Foundation for the situation calculus. *Electronic Transactions on Artificial Intelligence* 2(34), 159–178 (1998)
- 6 Dey, A.K., Abowd, G.D., Salber, D.A.: A Context-based Infrastructure for Smart Environment. In: *MANSE 1999*, pp. 112–128 (1999)
- 7 Lee, K., Kim, K.: Service Selection Model using Situation in Ubiquitous Computing Environment. In: *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance*, pp. 147–151 (2006)

Improving Trade-Offs in Bilateral Negotiations under Complete and Incomplete Information Settings

Ivan Marsa-Maestre, Miguel A. Lopez-Carmona, and Juan R. Velasco

Departamento de Automatica, Universidad de Alcala, Spain
{ivan.marsa,miguelangel.lopez,juanramon.velasco}@uah.es

Abstract. A bilateral negotiation may be seen as an interaction between two agents with the goal of reaching an agreement over a given range of issues which usually involves solving a conflict of interests between the agents. Usually, the agents taking part in the negotiation will consider different issues to be the most important ones for satisfying their goals, which allows to make issue trade-offs to search for joint gains. In particular, similarity criteria have been used to perform trade-offs in bilateral negotiations. This approach behaves differently depending on the knowledge each agent has about its counterpart, and depending on the order in which the different issues are considered. In this paper we propose two new approaches to improve the search for win-win solutions, one for complete information settings and the other for incomplete information settings. The experimental evaluation shows how our proposals improve the efficiency and optimality of the negotiation process over previous approaches.

1 Introduction

Integrative negotiation approaches intend to allow negotiating agents to search for joint gains when pursuing an agreement [1,2,3]. For this to be possible, a multi-issue negotiation scenario is required. In these scenarios, the negotiation process and outcome is determined by the participant's satisfaction functions and the impact that the different issues under negotiation have over the utility that each proposed contract yields to the negotiating agents. If the impact of the issues under negotiation over the satisfaction function is different for each agent (that is, some issues are more important for the player than for the opponent and vice versa), the issues may be traded-off against one another, increasing the social welfare of the deal [4]. In particular, this paper covers multi-issue bilateral negotiations, which involve a bargaining process between two agents (a *player* and an *opponent*), which exchange proposals (*contracts*) in order to reach an agreement over a given range of issues. For these scenarios, several heuristic approaches have been described, using different techniques. In [5], for example, similarity criteria are used to select contracts during an iterated hill-climbing search over the solution space. However, the random nature of the search impacts

the efficiency of the process, making necessary large populations of candidates to ensure the similarity-based selection is really effective. In this paper we show that the outcome and performance of the trade-off process may be improved by using information about the derivatives of the agent's valuation functions to direct the search for joint gains to the regions of the solution space where, according to the information available, it is more likely to find an optimal solution. In addition, we observe that the order in which the different issues are processed by the trade-off algorithm greatly impacts the final outcome. This is specially problematic in incomplete information scenarios [6], where there is no information about each agent's counterpart preferences that could be used to determine the best issue ordering for the algorithm. For these cases, we propose to use random permutations of the ordering of the issues at each iteration of the algorithm to mitigate the effect of the uncertainty about the opponent's preferences over the final outcome of the trade-off process. By performing experiments comparing the basic trade-off algorithm to our mechanisms we show how our proposals may provide benefits in terms of performance and optimality over previous works.

The rest of the paper is organized as follows. Section 2 recalls the most relevant previous works our research is related to. Section 3 describes the mechanisms we propose to improve trade-offs under complete information and incomplete information settings. The experimental evaluation is provided in section 4. The last section summarizes our main contributions and sheds light on some future research.

2 Similarity-Based Negotiation Trade-Offs

In [5], an algorithm for carrying out trade-offs in automated negotiations is proposed. It is based on finding a win-win solution through an iterated hill-climbing search in a landscape of possible contracts. Contracts are defined as sets of values for the different issues which are being negotiated, and agent satisfaction degrees for a given contract are computed using a weighted sum of monotonically increasing or decreasing scoring functions for each issue. Also, the concept of *iso-curve* is defined as the curve comprising the solutions which yield a given satisfaction degree for a given agent. The interaction protocol is a positional bargaining, that is, only specific solutions to the negotiation problem are exchanged between the agents. Once both agents participating in the negotiation have proposed an initial solution, solutions proposed by both agents in the subsequent steps of the negotiation are points lying in the same *iso-curve* while maximizing the similarity to the opponent's last offering. At any given point of the negotiation, the search of the next proposal to make is performed by successively generating random contracts which lay closer to the *iso-curve* and selecting the most similar contract to the opponent's proposal. The algorithm terminates when the last selected contract lies in the *iso-curve*. This selected contract is sent to the opponent, which runs the algorithm to generate the next proposal. The process continues until the proposal generated by an agent is accepted by its counterpart.

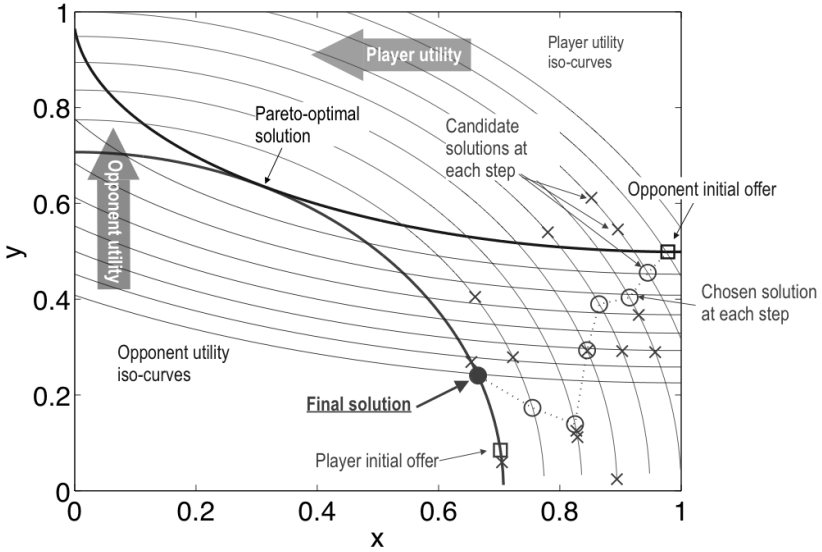


Fig. 1. Example of the trade-off algorithm with $N=3$ and $S=7$

Figure 1 shows a conceptual example of the algorithm for a negotiation regarding two issues, x and y . The algorithm starts at the opponent’s last offer, and moves toward the iso-curve associated with the player’s last offer. This movement is performed in S steps. At each step, N children contracts are generated which are closer to the player’s target iso-curve by a utility difference E . That is, all generated children lie in a player iso-curve which yields an increment E in utility to the player compared to the children generated at the previous step. From all generated children contracts, the most similar to the opponent’s proposal is chosen as the starting point for the next step. After S steps, the chosen contract lies in the target iso-curve, so the algorithm terminates and the chosen contract is sent to the opponent as the next proposal.

3 Improving the Trade-Off Algorithm under Complete and Incomplete Information Settings

We define the issues under negotiation as a finite set of variables $x = \{x_i | i = 1, \dots, n\}$, and a contract (or a possible solution to the negotiation problem) as a vector $s = \{x_i^s | i = 1, \dots, n\}$ defined by the issues’ values. The overall (or global) satisfaction degree of a potential solution s for an agent j is $V^j(s) = \oplus \{V_i^j(x_i^s) | i = 1, \dots, n\}$, where \oplus is an aggregation from $[0, 1]^n$ to $[0, 1]$, and $V_i^j(x_i)$ is the agent j ’s scoring function for the issue x_i . For this work we restrict ourselves to weighted additive aggregation functions and independent scoring functions for each issue in the negotiation. That is, the overall satisfaction degree of a potential solution s for an agent j is $V^j(s) = \sum_{1 \leq i \leq n} \omega_i^j V_i^j(x_i^s)$, where $W^j =$

$\{\omega_i^j | i = 1, \dots, n\}$ models the importance that agent j assigns to each decision variable i under negotiation as a weight ω_i^j . Within this framework, we define two mechanisms to improve the trade-off algorithm. The first one uses a derivative-based approach to direct the search for solutions to a region of the solution space where is easier to find an agreement. The second one uses random permutations to mitigate the impact that the order in which the algorithm processes the issues has over the outcome of the negotiation. Intuitively, we can see that the latter mechanism will be more suitable for incomplete information scenarios (where the uncertainty about the best issue ordering is higher), while the former will be applicable in complete (or information scenarios, where information about each agent's valuation function is available to its counterpart.

3.1 Using Derivatives within the Trade-Off Algorithm

The trade-off algorithm [5] performs an iterated hill-climbing search over the solution space. This is done by starting at the opponent's last proposal, y , and moving towards the iso-curve associated with the agent's target increase in utility E . The algorithm performs a total of S steps, and at each step it generates N children contracts which are closer to the iso-curve than the ones in the previous step. From all the children, the most *similar* to the opponent last proposal is selected as the starting point for the next step. The algorithm generates children by splitting the gain in utility randomly among the set of issues under negotiation. For each issue i , the algorithm assign an utility increase for this issue $r_i = \min(\text{random}(E_i), \frac{E-E_n}{\omega_i})$, where E_i is the maximum gain for the issue x_i at this step, and $\frac{E-E_n}{\omega_i}$ is used to limit the final gain to E . Since the generation of childrens at each step of the hill climbing process is random, a mechanism which may be used to increase the effectiveness and efficiency of the search for solutions is to perform a more *directed* hill-climbing, that is, to generate the children at each step in the direction that causes the least satisfaction loss to the opponent while increasing the agent's own utility.

To this end, we allow an agent to use *gradient information* to influence the hill-climbing path followed to generate the intermediate solutions for the different steps of the trade-off algorithm. The information used is defined as a vector $d_{req} = \{d_i | i = 1, \dots, k; k < n\}$, where d_i is computed by normalizing the partial derivatives $\frac{\partial V(s)}{\partial x_i}$ of the global satisfaction function of the agent issuing the request at the point defined by its opponent proposal. What we propose is using this information to modulate the random utility gain splitting computed in each iteration of the trade-off algorithm. Since the partial derivatives of the scoring functions express how an agent's utility varies with the variation of each individual issue, this information may be used to weigh the utility increase for each issue at each step, so that the utility increase is performed mainly over the attributes that less impact the other agent's utility. The point in the algorithm to perform this modulation is when the algorithm assigns an utility increase r_i for each issue x_i . The utility increase is defined as $r_i = \min(\text{random}(\frac{E_i}{d_i}), E_i, \frac{E-E_n}{\omega_i})$, thus assigning more utility gain to those issues where the partial derivatives $\frac{\partial V(s)}{\partial x_i}$

express a lesser impact over the opponent’s utility and vice versa. Algorithm **1** shows the modified trade-off algorithm to take advantage of gradient information.

Our hypothesis is that using derivatives in this way within the algorithm will direct the hill-climbing process to solutions that, while keeping the player’s satisfaction constant, have a lesser impact over the opponent’s satisfaction, thus improving the outcome of the trade-off algorithm in terms of player and opponent’s utility. Furthermore, by restricting the hill-climbing path to a direction known to provide more satisfying solutions, less children will be needed to achieve a certain result, thus improving algorithm efficiency in terms of computational complexity.

Input:

- y : last step opponent’s offer
- E : step utility increase
- $V_i()$: Value scoring functions for the decision variables
- ω_i : importance weights for the decision variables
- d_i : normalized derivative of the opponent’s scoring function for the decision variables

Output: Y : new child of y

```

foreach decision variable  $i$  do
  |  $E_i = 1 - V_i(y_i)$ 
end
 $k = 0; E_n = 0;$ 
while  $E_n < E$  do
  |  $k = k + 1$ 
  | foreach decision variable  $i$  do
  | | if  $E_n < E$  then
  | | |  $r_i = \min(\text{random}(\frac{E_i}{d_i}), E_i, \frac{E - E_n}{\omega_i})$ 
  | | | else  $r_i^k = 0$ 
  | | | end
  | | end
  | end
  | foreach decision variable  $i$  do
  | |  $E_i = \sum_{1 \leq j \leq k} r_i^j$ 
  | |  $Y = V_i^{-1}(V_i(y_i) + E_i)$ 
  | end

```

Algorithm 1. Children generation for the trade-off algorithm using gradient information

3.2 Using Random Permutations of the Issue Ordering

Though children generation within the trade-off algorithm is based on randomly splitting the gain in utility among the different issues, this random energy distribution is not uniform, and thus generated children are not uniformly distributed either. Each one of the N children used at each one of the S steps is generated as follows. The maximum gain in utility for each issue is $E_i = 1 - V_i(y_i)$, and the

maximum overall utility gain for each step is $E = \frac{V(x)-V(y)}{S}$, computed by dividing the difference in player's utility for the agents' last proposals between the number of steps. The algorithm cycles through the different decision variables, assigning a random utility gain to each one of them until the total increase of utility E_n reaches E . To make sure that neither the maximum utility gain per issue E_i nor the maximum overall utility gain E are exceeded, the utility increase for each issue at each cycle k is defined as $r_i^k = \min(\text{random}(E_i), \frac{E-E_n}{\omega_i})$, and the values of E_i and E_n are updated after each utility assignment. On average, and assuming perfect random number generation, each utility assignment will increase E_n by half of the issue's remaining potential gain $\omega_i E_i$. Therefore, issues processed by the algorithm first will, on average, be assigned higher utility gains, and thus children will be unevenly distributed in the solution space.

We can see how this bias in children generation at each step may affect the final outcome of the trade-off algorithm in terms of joint utility. Let p and o be the player and opponent agents in a negotiation, respectively, and let x and y be the last proposals from the agents. The player uses the trade-off algorithm to hill-climb from the opponent's last proposal to a solution s which gives her the same utility that her own last proposal. Therefore, the player's utility after the trade-off will always be $V^p(s) = V^p(x)$, and thus the evaluation of the algorithm effectiveness must be based on the opponent's utility $V^o(s)$. In a worst case scenario, the agent's preference functions V_i^j for each issue are such that an increase of the utility for a player in a given issue causes the utility for the opponent in this issue to decrease in the same amount. The impact of varying issue i utility over the overall utility gain for each agent is determined by the weights ω_i^j . In particular, $\Delta V^o = -\frac{\omega_i^o}{\omega_i^p} \Delta V^p$ when only issue i is varied. Therefore, the trade-off will be more effective if higher utility gains are assigned to the issues minimizing $\frac{\omega_i^o}{\omega_i^p}$. Under complete information settings, this may be achieved by ordering the cycling through the issues according to $\frac{\omega_i^o}{\omega_i^p}$, making the algorithm process first the decision variables which are more important to the player than to the opponent. However, under incomplete information settings, the opponent's weights may not be known to the player. In this cases, using an arbitrary issue ordering may negatively impact the outcome of the negotiation. To address this problem, we propose to use a different random ordering of the issues to generate each children at each step. In this way, the N children generated at a given step of the algorithm will be uniformly distributed in the solution space, and similarity criteria may be used to choose among children without a bias due to issue ordering.

4 Experimental Analysis

Our experiment plan is designed to determine whether the proposed mechanisms provide an improvement to the efficiency and optimality of the negotiation process over the previous work described in Section 2. To this end, we have reproduced the experiments performed in [5], comparing the results of their trade-off algorithm with the results obtained applying the proposed mechanisms.

4.1 Experimental Settings

To evaluate the contribution of the proposed mechanisms to the trade-off algorithm, *single offer experiments* have been performed. The experimental procedure consists of inputting two contracts (representing the agent’s initial utterances) into the algorithm and observing the execution trace of the algorithm for *one* offer from the *player* to the *opponent* (i.e. observing how the algorithm climbs from the opponent’s proposal to a new proposal which have the same utility than the player’s initial proposal in S steps). Contracts are chosen so that they give a high utility to the proposer and a low utility to its counterpart. The importance weight vectors of the agents, used to compute the global satisfaction function for each agent, are fixed throughout the negotiation: $W_{player} = [0.15, 0.25, 0.1, 0.5]$ and $W_{opponent} = [0.35, 0.05, 0.5, 0.1]$.

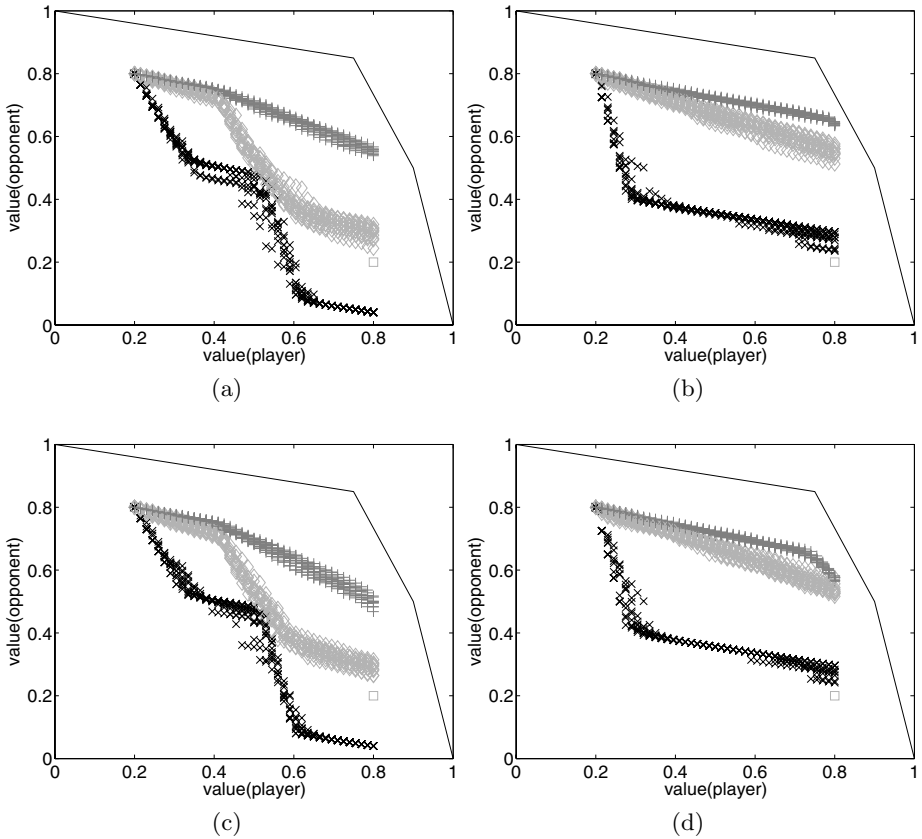


Fig. 2. Effect of the use of knowledge about derivatives ($W_{opponent}$) over the trade-off algorithm under different issue orderings: a) all derivatives, order [1 2 3 4], b) all derivatives, order [3 2 4 1], c) two derivatives, order [1 2 3 4], d) two derivatives, order [3 2 4 1]

The effect of derivatives has been measured performing different experiments, ranging from assuming knowledge about the derivatives for one issue to assuming perfect knowledge about the derivatives for all issues. The outcome of the experiments has been compared to the same experiments without using derivatives. Finally, taking into account that, as observed in [7], the order in which the different issues are processed by the trade-off algorithm greatly impacts the final outcome, we have repeated the experiments for different issue orderings.

The effect of using random issue-ordering permutations over the trade-off algorithm has been measured by applying random permutations as defined in Section 3.2, and comparing the outcome to the same experiment made with random *stationary* issue orderings, that is, orderings which are chosen at the beginning of the algorithm execution and do not change between algorithm iterations.

To evaluate the contribution of the different mechanisms to the algorithm in terms of effectiveness, we have performed the experiments for the best case described in [5], using $S = 40$ as the number of steps to reach the iso-curve and $N = 100$ as the number of children generated at each step. To evaluate the contribution of the different mechanisms to the algorithm in terms of efficiency, we have performed a set of experiments varying the number of children N , in order to test our hypothesis that fewer children are needed to achieve the same result when using derivatives or permutations within the trade-off algorithm.

4.2 Experimental Results

Figure 2 shows the results of using derivatives within the trade-off algorithm. Each graphic shows the results of 10 runs of the experiment. The x-axis and y-axis represent, respectively, the *player* and *opponent* utilities. A black line joining (0,1) and (1,0) represent the Pareto-optimal line, computed using the weighted method [4,8], so that the optimality of the heuristic approaches may be assessed against the analytical optimum. For each run we have represented the initial contracts issued (depicted as gray squares), and the execution trace of the trade-off algorithm under evaluation. The points represent the hill-climbing paths followed by the algorithm from the opponent initial contract (upper left corner) to the player's trade-off proposal (right side of the graph). The trace of the basic trade-off algorithm has been represented using light gray diamonds, while the trace of our proposed derivative-based approach has been represented using dark grey plus signs (+). For comparison, a random reference trade-off

Table 1. Statistical results for 100 runs of the experiment comparing the basic trade-off algorithm with the use of derivatives

Figure	Basic		Derivatives	
	median	conf. interval	median	conf. interval
1. (a)	0.2844	[0.2807, 0.2881]	0.5528	[0.5499, 0.5556]
1. (b)	0.5411	[0.5375, 0.5488]	0.6404	[0.6395, 0.6413]
1. (c)	0.2867	[0.2828, 0.2905]	0.4921	[0.4898, 0.4945]
1. (d)	0.5430	[0.5390, 0.5469]	0.5747	[0.5735, 0.5759]

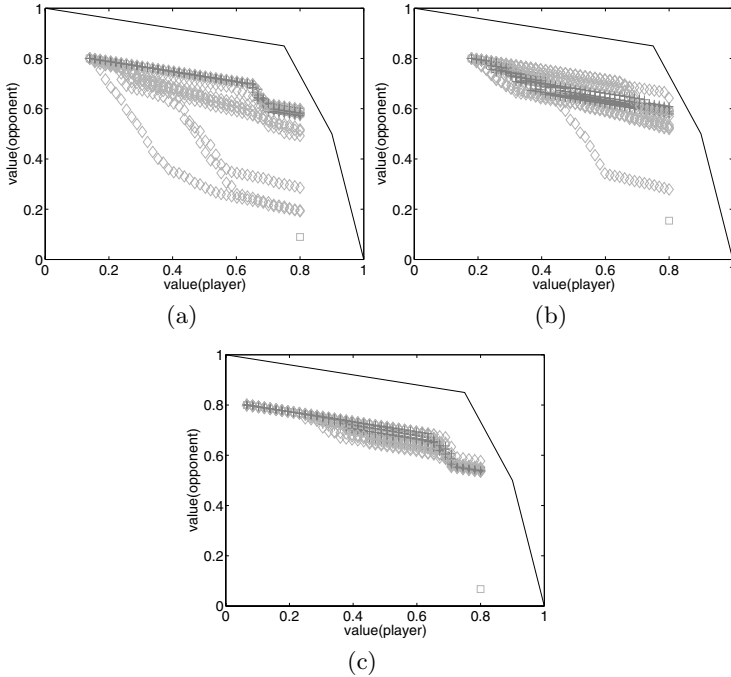


Fig. 3. Effect of the use of random permutation in children generation within the trade-off algorithm. a) permutation applied to all issues, b) permutation applied to three issues, c) permutation applied to two issues.

algorithm –where candidates at each step are randomly chosen among all generated children without using similarity criteria or derivatives– has been represented using black crosses (x). In figures 2 a) and b) we have used information about *all* the partial derivatives of the opponent’s valuation functions, which is the same knowledge used in 5 for the perfect knowledge experiments, since for a linear additive scoring system, the opponent weights $W_{opponents}$ equal the partial derivatives of the valuation function. In figures 2 c) and d), we have assumed knowledge of the partial derivatives for only two issues. Table 1 shows the medians and the 95% confidence intervals for 100 runs of the experiments depicted in the figure. We can see that there is a significant improvement of the utility of the final outcome for the opponent, and that the improvement is more significant for some orderings, yielding utility gains of nearly 80% over the approach in 5. We can also see that the results achieved using derivatives are closer to the Pareto-optimal line than those obtained by using the basic similarity-based approach. From these results we can conclude that the use of derivatives makes the trade-off algorithm more robust to the ordering of the issues. In addition, we can see that using incomplete knowledge of the derivatives makes the improvement decrease, but still keeping the results better to the ones yielded by the basic algorithm. This shows that the derivative based approach is not only

Table 2. Statistical results for 100 runs of the experiment comparing the basic trade-off algorithm with the use of permutations

Figure	Basic		Permutation	
	median	conf. interval	median	conf. interval
2. (a)	0.304	[0.2541, 0.3539]	0.5546	[0.5538, 0.5554]
2. (b)	0.5397	[0.4899, 0.5776]	0.6053	[0.6009, 0.6096]
2. (c)	0.5648	[0.5611, 0.5684]	0.5620	[0.5605, 0.5635]

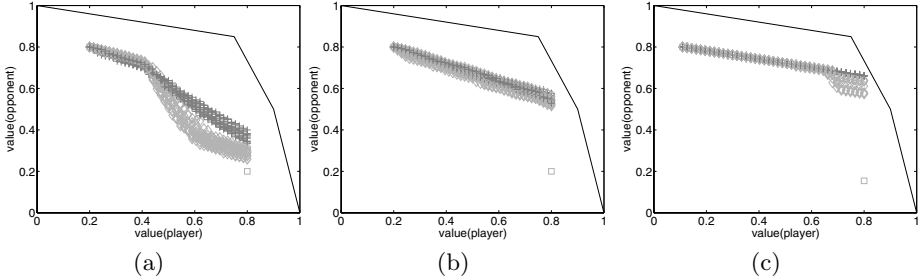


Fig. 4. a) and b): Comparison of the outcome of the basic tradeoff algorithm with $N = 100$ against the use of derivatives with $N = 10$ for ordering [1 2 3 4] (a) and [3 2 4 1] (b). c): Comparison of the outcome of the basic tradeoff algorithm with $N = 100$ against the use of random permutations with $N = 10$. Permutation applied to two issues.

useful in complete information settings, but also in settings where there is partial information available about the opponent’s valuation function.

Figure 3 shows the results of using random ordering permutations for children generation as described in section 3.2. Each graphic shows the results of 10 runs of the experiment. Figure 3 a) shows the results when random permutation is applied to all issues. Figure 3 b) shows the results obtained when the algorithm takes the issue which minimizes $\frac{\omega_i^o}{\omega_i^p}$ first, and the rest of issues are randomly ordered. Finally, figure 3 c) shows the results when the two issues which minimize $\frac{\omega_i^o}{\omega_i^p}$ are processed by the algorithm first and random permutation is applied to the remaining two issues. Table 2 shows the medians for the results obtained. We can see that, in the case of uncertain knowledge (all issues are ordered randomly) the results of the basic trade-off algorithm have a higher variance, while the utility yielded by the random permutation approach is more stable. On average, using random permutations for children generation yields a significant improvement in opponent’s utility when no information about the opponents weights is known, and the improvement decreases when more information is available. We can see that, when there are only two issues whose ordering is unknown (Figure 3 c), using random permutations yields slightly worse performance than the basic algorithm, since there are only two possible orderings. Therefore, random permutations should only be applied when there is uncertainty about the best ordering of the issues.

Finally, Figure 4 compares the derivative-based and permutation-based approaches to the basic trade-off algorithm in terms of efficiency, showing the results when using different number of children in the algorithm. We can see that both approaches can yield to a significant reduction of the number of children needed for a given outcome, thus increasing the efficiency of the negotiation process. Figure 4 shows that, using the proposed mechanisms, we can reduce the number of children by even one order of magnitude while achieving the same effectiveness. Since, for a single offer experiment, the algorithm is called SN times, where S is the number of steps used for the algorithm and N the number of children generated as each step, this reduction greatly enhances the efficiency of the process.

5 Conclusions and Future Work

In automated multi-issue negotiation scenarios, it is not unusual that the agents taking part in the negotiation will consider different issues to be the most important ones for satisfying their goals. The difference between the weights that each negotiating agent gives to each issue allows negotiators to make trade-offs, varying the values of the different issues so that the utility for one agent may remain the same while the utility for its counterpart increases, thus improving overall joint utility. In [5], similarity criteria are used to perform trade-offs in bilateral negotiations. In this paper, we present two different mechanisms to improve the similarity-based trade-off algorithm. The first is based on using knowledge about the derivatives of the opponent's valuation functions to influence the direction in which new solutions are searched for. The second uses random issue permutations for children generation to make the algorithm more robust against the uncertainty about the correct issue ordering. Our experiments have validated our hypotheses: that the proposed mechanisms improve the basic trade-off algorithm both in terms of optimality and performance. The derivative-based approach is more suitable for complete information scenarios (where information about the opponent's valuation functions is available), while random permutations should be used for incomplete information settings [9], where the correct issue ordering is not known.

Though the experiments have yielded satisfactory results, there is still plenty of research work to be done in this area. Metastrategy experiments as defined in [5] should be performed to evaluate the contribution of the proposed mechanisms to the overall negotiation process. A more in-depth performance analysis of the algorithm is main priority for future work. Finally, we are interested in extending the trade-off algorithm and our mechanisms to make them able to handle nonlinear scoring functions and issue interdependency, which are the real challenges in complex negotiation.

Acknowledgment

This work has been supported by the Spanish Ministry of Education and Science grant TSI2005-07384-C03-03 and the Comunidad de Madrid grant CCG07-UAH/TIC-1648.

References

1. Lopez-Carmona, M.A., Velasco, J.R., Marsa-Maestre, I.: The agents' attitudes in fuzzy constraint based automated purchase negotiations. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS 2007. LNCS, vol. 4696, pp. 246–255. Springer, Heidelberg (2007)
2. Klein, M., Faratin, P., Sayama, H., Bar-Yam, Y.: Protocols for negotiating complex contracts. *IEEE Intelligent Systems* 18(6), 32–38 (2003)
3. Ito, T., Klein, M., Hattori, H.: A multi-issue negotiation protocol among agents with nonlinear utility functions. *Multiagent and Grid Systems* 4(1), 67–83
4. Raiffa, H.: *The Art and Science of Negotiation*. Harvard University Press (1982)
5. Faratin, P., Sierra, C., Jennings, N.R.: Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence* 142(2), 205–237 (2002)
6. Jonker, C., Robu, V.: Automated multi-attribute negotiation with efficient use of incomplete preference information. In: *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pp. 1054–1061 (2004)
7. Ros, R., Sierra, C.: A negotiation meta strategy combining trade-offs and concession moves. *Autonomous Agents and Multi-Agent Systems* 12(2), 163–181 (2006)
8. Ehtamo, H., Ketteunen, E., Hamalainen, R.P.: Searching for joint gains in multi-party negotiations. *European Journal of Operational Research* 1(30), 54–69 (2001)
9. Lai, G., Li, C., Sycara, K.: Efficient multi-attribute negotiation with incomplete information. *Group Decision and Negotiation* 15(5), 511–528 (2006)

PAMS – A New Collaborative Framework for Agent-Based Simulation of Complex Systems

Trong Khanh Nguyen², Nicolas Marilleau¹, and Tuong Vinh Ho²

¹ Geodes, Institut de Recherche pour le développement (IRD),
32 av. H. Varagnat, 93143 Bondy Cedex, France

² MSI Lab, Institut de la Francophonie pour l'Informatique (IFI),
ngo 42, Ta Quang Buu, Ha Noi, Viet Nam
ntkhanh@ifi.edu.vn, nicolas.marilleau@ird.fr,
ho.tuong.vinh@auf.org

Abstract. Major researches in the domain of complex systems are interdisciplinary, collaborative and geographically distributed. The purpose of our work is to explore a new collaborative approach that facilitates scientist's interactions during the modelling and simulating process. The originality of the presented approach is to consider models and simulators as a board of the collaboration: a shared object manipulated by a group of scientists. Agent-based simulations are powerful tools for studying complex systems. In this context, we develop a collaborative platform dedicated to agent-based simulation (PAMS). This new environment integrates common collaborative tools (e.g. videoconferencing, instant messaging, whiteboard) and specific tools to share and manipulate models, simulators, experiments and results... The current version of PAMS is based technologies coming from distributed systems. Today PAMS has been designed to support major agent based simulation frameworks. This paper aims to give an overview of the PAMS environment by defining the collaborating approach, the framework architecture and an example of its utilization.

Keywords: Collaborative simulation, agent-based simulation, distributed systems.

1 Introduction

Modeling and simulation often requires cooperation between researchers from different disciplines. Data collection, model conceptualization and implementing those models using computational tools all require close teamwork amongst various players (domain experts, modelers and computer scientists). Modern-day research projects are interdisciplinary, collaborative, and researchers are often geographically separated. Given these modern conditions, the use of collaborative systems becomes essential to facilitate complex systems research when a team of geographically and professionally dispersed researchers must work together towards a common goal.

The advent of new information technologies and communications tools over the past fifteen years has enabled the development of a plethora of collaborative platforms [2]. A few of them, such as BSCW [11], E-Groupware[12], and Sakai [1] have

positioned themselves as collaboration-oriented extensions of generalist communications platforms in the scientific world. These products integrate specific functionalities facilitating: (i) access to knowledge and scientific information, (ii) interaction and collaboration between researchers, and (iii) a more effective dissemination of research results. These activities are often referred as “E-Research”. The platforms supporting such work are referred to as Virtual Research Environments (VRE) [13].

VRE systems are still relatively undeveloped. Most of the available environments remain incapable of supporting significant collaborative efforts; despite growing demands for such tools the scientific community [6]. Most of these collaboration systems are simply data repositories with web interfaces [5].

In the domain of modeling and simulation of complex systems, the use of agent-based simulation models (ABM) is increasingly popular. Many ABM platforms (Repast [8], NetLogo [9], Swarm [7] or GAMA [3]) have been developed and used by researchers [3, 10]. In the context of collaborative research, one major question is: How can geographically diverse researchers effectively work together to conduct ABM simulations without regard to the ABM platforms used?

One of our research interests focuses on the design and implementation of collaborative environments for computer modeling and the simulation of complex systems, essentially those based in ABM. The main idea is to place models, simulators, experiments and results at the center of the collaboration. From this idea, we have designed and developed methodologies and a form of “groupware” (known as “PAMS”) for supporting collaboration between domain experts, modelers and computer scientists. PAMS is a type of web-based groupware containing common collaborative tools (video-conferencing, instant messaging, and so on) and specific tools dedicated to the simulation domain (sharing experiments, results, experience exchange...).

The aim of this paper is to present the PAMS framework. Firstly, we shall summarize the platform’s functionalities. Secondly, we provide a short description of the PAMS architecture. Finally, a case study is described demonstrating how a collaborative simulation might be executed within this new paradigm.

2 PAMS – A New Collaborative Framework for Agent-Based Simulation of Complex Systems

The PAMS project introduces a new approach to collaboration in research projects. Models and simulators will no longer simply come in the form of “research results” distributed to the community through scientific communications (journal articles, workshops etc.). Rather, with PAMS, models and simulators become concrete entities available to interested parties on the web to support collaborative work and research. The originality of our approach is to consider the model or the simulator as an object shared by a group of researchers, which can be manipulated, configured, analysed and so on.

PAMS groupware is an environment allowing researchers to work together in designing or exploring models (execution of models based on various scenarios, interactions conceptualized in various contexts or of a specific simulation). For example, consider a scenario for collaboration between researchers in which they used the software environment in development.

Scenario1: “Collaborative experimentation from distance”

Mr. X and Y, researchers in ecology, respectively located in Paris and Hanoi, wish to execute some simulations in order to get results that will be illustrated in an article they will write together. The two researchers connect to the project website and begin a private discussion. Mr. X starts the simulator, and suggests that Mr. Y. shares his interface. Mr. X starts entering simulation parameters, which are not suitable to Mr Y. Mr Y starts a videoconferencing session to share with Mr. X his surprise. He posts a note on the simulator interface indicating the value of parameters that seem correct to him. After discussion, and display, by Mr. Y, part of the item they want to illustrate, the two fall into agreement on common values. Mr. X launches the simulation and then chooses to display only the graph of the evolution of biodiversity, while Mr. Y visualizes the spatial distribution of species. Each researcher posts annotations of visualization in real time on his own interface, which also appear on the interface of his colleague and they begin holding a discussion (registered as such in the instant messaging system). Due to time constraints, however, Mr. X must leave. They both decide to resume this discussion later and record the session. Three days later, returning to the site, the conversation restarts and resumes in the state where it had been left earlier...

This scenario shows that the use of a collaborative tool firstly: addresses the problem posed by the geographical dispersion of researchers and secondly, brings a fresh dimension to the simulation activities of complex systems.

An agent-based simulator is run many, many times using different parameter values in order to postulate various situations and to understand the dynamics of the studied system. In this context, users focus on inputs and outputs of a model. They are free to forget how the model works. For this reason, we assume that a perfect agent-based simulator is a “black box” that scientists (except for the box’s creators) use by defining inputs and analyzing outputs without really caring how the “black box” works.

Existing Open Source web-based groupware constitutes a solid basis upon which it is possible to add new modules providing collaboration in the field of complex systems. We chose one of them called “Sakai”[1] and added new collaborative modules dedicated to simulation activities, such as: (i) setting simulations; (ii) executing simulations on a remote server; (iii) visualising and analysing results; (iv) managing versions of available models; (v) archiving experiments and results; (vi) annotating experiments and results (giving contextual comment).

3 PAMS: A Modular Environment

3.1 Logical Architecture

PAMS is based on a multi-tier architecture called Model View Controller (MVC) [15]. This approach distinguishes graphical user interface (Presentation Tier) from the kernel of the application (Logic Tier) and databases (Data Tier).

Fives modules compose the kernel of the PAMS environment (See figure 1):

–**Simulation platform drivers package** contains the kernel of agent-based platforms such as Repast [8] or Gama [3].

- Outputs package** manages simulation results coming from simulation platforms, and shares these data for other modules of the systems.
- Displays package** formats shared outputs in order to generate and manage user displays: monitors, plots and/or 2D grids (images).
- Recorder package** saves every value that an output has taken during a simulation in a database. This data is read by the experiment’s web browser interface.
- Controllers package** aims at managing experiments and simulators, ensuring the coherency and concurrency of objects shared by users (parameters, simulation outputs, experiments...).

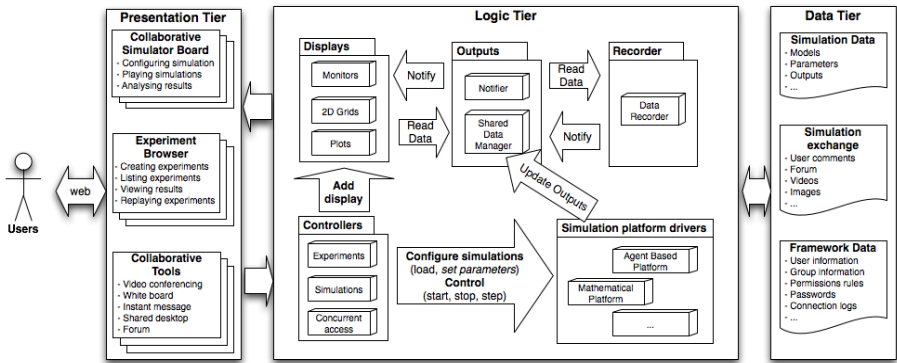


Fig. 1. PAMS logical architecture

The PAMS environment proposes a generic web-based collaborative GUI. This interface takes advantage of typical collaborative tools (video-conferencing, white-board and so on) coming from Sakai and Agora tools. In addition, PAMS provides functions dedicated to the simulation domain:

- A collaborative simulation board** for executing and sharing remote simulators
- An experiment browser** for managing and replaying completed experiments and exchanging results.

Today, PAMS supports simulators derived from two agent-based platforms: Repast and Gama. A few famous and simple simulators were deployed, e.g. the life game (Enn for Repast, Life for Gama) to test PAMS functionalities. In addition, specific simulators, as GamaAvi, with its origins in a multi-disciplinary research project, are and will be added. Scientists (epidemiologists, geographers, computer scientists or mathematicians) will use these simulators to run experiments and to collaborate.

3.2 Technological Architecture

The PAMS framework is a distributed system. It can be viewed as a container in which simulators are loaded, connected with a database, executed by dedicated servers and managed through a web interface.

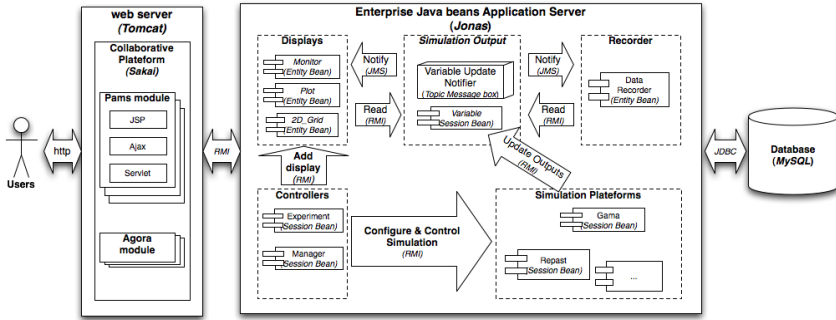


Fig. 2. Technological architecture

PAMS is based on common technologies from the domain of distributed systems (see figure 2):

- A **web application server** based on Jsp, Ajax and Servlet (*Tomcat*) to manage the collaborative and adaptable web interface that displays models, experiments, simulators and results with a simple, adaptable and didactic structure of data.
- An **Enterprise Java bean application server** (*Jonas*) to execute simulators, manage experiments, inputs, outputs and collaboration.
- A **Database** (*MySQL*) to store models, experiments inputs, outputs and exchanges between PAMS users.

The PAMS framework takes advantage of the Sakai environment [1]. Sakai is an online Collaboration and Learning Environment that permits research exchange. From the Sakai environment, we developed new modules that manage a collaborative GUI specific to the simulation domain (experiment viewer, simulation board etc). The PAMS module is associated with the Agora environment [14]. Agora is a plug-in of Sakai that offers typical collaborative features like videoconferencing, whiteboard, chat and others.

The use of Enterprise Java Beans is one of the keys that allows PAMS environment to be flexible, modular and modifiable. Each module of the framework is composed of several EJBs. Every EJB of a same module is used through a unique interface (determined for the module). To improve the PAMS framework, new EJB could be developed and dynamically deployed without revising old PAMS components. But, these new EJBs must follow predefined interfaces of the PAMS.

The PAMS environment can be deployed on a GRID of computers. Thanks to this distributed architecture, load-balancing rules can be imagined to spread experiment executions over a GRID.

Most of agent-based simulators depend on a specific platform such as Netlogo, Repast or Madkit. These agent-based frameworks must be installed on the computer before the setup of the simulator. Sometimes, simulators need a database to obtain working data. Simulator setups are not trivial: much experience in computer science is required. In addition, many simulators need resources (memory or processor) that are not available on a personal desktop or laptop. Thanks to the web interface of PAMS

and its distributed architecture, scientists have nothing to install before using the framework. In addition they can take advantage of resources provided by the Grid in which the platform is executed.

4 Case Study

The aim of this case study is to show how a group of scientists can use the PAMS environment to run simulations and collaborate. Scenario 1 (“Collaborative experimentation from a distance”) can be taken as an example. In this instance, Mr. X and Mr. Y, want to study the famous Life game model [4].

Consider that a PAMS service is running on a server. This service supports various agent based simulation frameworks, in particular the Repast environment. Several simulators have been installed, deployed and are available. For example, the environment proposes the Enn simulator, which is a Repast version of the Life game model.

Using a web browser, Mr. X and Mr. Y connect themselves to the PAMS platform. After the identification step, they access to their private workspace. Mr. X is the initiator of the experiment. He has to create an experiment and to determine the participants. From a list available in the agent-based simulator, Mr. X selects the Enn simulator. A new display appears that shows information about the Enn simulator: aims of the model, inputs, outputs and so one. On this screen, Mr. X can see every public experiment done with the Enn simulator. But, he prefers to create a new one. For that, X inputs a comment about the new experiment (its aim) and selects participants from a list of subscribed persons. In the case of this scenario, Mr. X selects Mr. Y. and submits the form. The experiment is now created. X is waiting Y’s connection to start simulations.

Y selects the Enn simulator from the list that contains available agent-based simulators. Information about this simulator is displayed, and Y sees that X has invited him to participate in an experiment. Y selects connects to the experiment.

X and Y are seeing the same display: the simulation board of the Enn simulator. For that, he takes the token. Mr. Y’s display is freezing. Y cannot perform the action on the simulator board, but he sees modifications. After doing modifications, X releases the token.

Y does not agree with X’s parameter modification. To explain his surprise, Y starts a videoconferencing session integrated in the Agora meeting tool (see figure 2). Thanks to video, audio and whiteboard tools, Y discusses with X. X and Y exchange their opinions through a user-friendly GUI. To illustrate its says, Y shows an article by sharing its desktop and convinces X. X wants to modify simulation parameters according to Y’s recommendation. He takes the token and changes parameter values. Then, X starts the simulation.

X and Y see the evolution of predetermined outputs in real time. During the simulation, they discuss the evolution of the outputs. From these results, X and Y begin an analysis and make hypothesis about the phenomena they see.

Due to time constraints Mr. X must leave. They both decide to resume this discussion later. So they record the session. Three days later, returning to the site, the conversation starts and resumes in the state where it had been left earlier...

Figure 3 shows the screen shots (of the PAMS environment) during the execution of the above scenario.

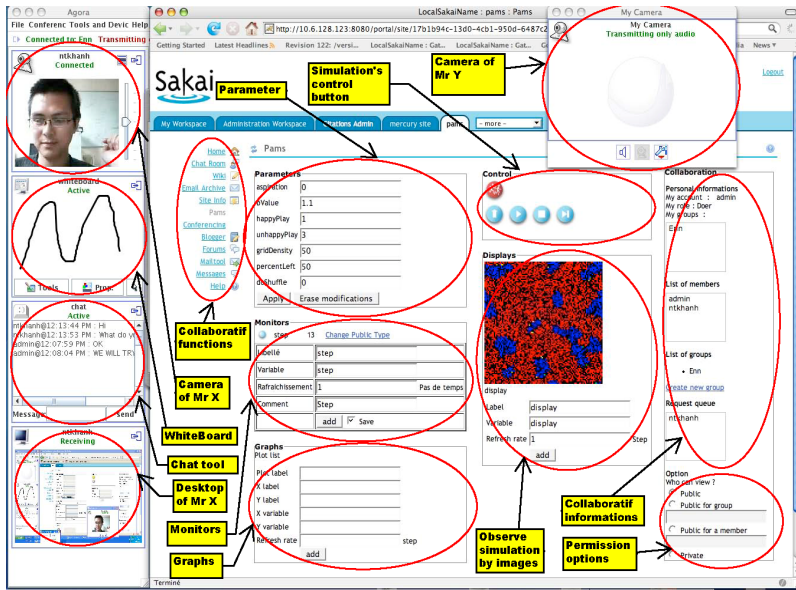


Fig. 3. Screen shot of the PAMS environment (Desktop of Mr. Y)

5 Conclusion

PAMS is a collaborative framework for simulating Agent-based models of complex systems. It is based on an original approach in which the simulator supports the exchange: it is a shared object manipulated by every member of a scientific group. In addition, we consider that users view a simulator as a “black box”. So users concentrate and collaborate on the input and output of the simulators.

Scientists are able to collaborate in this manner thanks to a web-based GUI that allows remote, shared access to simulators. This GUI contains common tools (video-conferencing, instant-messaging, whiteboard, and so one), which are improved by specific tools dedicated to the domain of the simulation of complex systems. This GUI supplies collaborative functions to setup simulators, to execute simulators on a remote server, to visualize and analyze simulation results, and to keep logs of each experiment.

Currently, the PAMS environment supports two agent-based simulation platforms (Repast and Gama). Adding new drivers will support of every simulation platform in the future. The modularity of PAMS permits our environment to be improved in many ways. Adding new kinds of displays or collaborative tools is one example.

The PAMS environment will be improved in many ways: (i) adding new collaborative tools (e.g. an annotating system to comment experiments), (ii) optimizing the system (e.g. adding load-balancing strategies), and (iii) supporting new agent based simulation platforms (e.g. Madkit, Repast and so on). Before that happens, we will have to test the existing version on a concrete research project applied to, for instance, geography or epidemiology. Feedback from these tests will provide the vital keys we require to further development and future improvements of the PAMS environment.

References

1. Severance, C., Hardin, J., Golden, G., Crouchley, R., Fish, A., Finholt, T., Kirschner, B., Eng, J., Allan, R.: Using the Sakai collaborative toolkit in e-Research applications. *Concurrency and Computation: Practice and Experience* 19(12), 1643–1652 (2007)
2. Saint-Voirin, D.: Contribution à la modélisation et à l'analyse des systèmes coopératif: application à la e-maintenance. Université de Franche-Comté, Besançon (2006)
3. Amouroux, E., Quang, C.T., Boucher, A., Drogoul, A.: GAMA: an environment for implementing and running spatially explicit multi-agent simulations. In: *Prima-2007*, Bangkok (2007)
4. Conway, J.: The Game of Life. *Scientific American* 223, 120–123 (1970)
5. Henriksen, J.O., Lorenz, P., Hanisch, A., Osterburg, S., Schriber, T.J.: Web based simulation center: professional support for simulation projects. *Winter Simulation Conference-2002* 1, 807–815 (2002)
6. Ahmed, K., Brahim, B.: Towards a Web Based Simulation Groupware: Experiment with BSCW. *Information Technology Journal* 1812(5638), 332–337 (2008)
7. Terna, P.: Simulation Tools for Social Scientists: Building Agent Based Models with SWARM. *Journal of Artificial Societies and Social Simulation* 1(2) (1998)
8. North, M.J., Collier, N.T., Vos, J.R.: Experiences Creating Three Implementations of the Repeat Agent Modeling Toolkit. *ACM Transactions on Modeling and Computer Simulation* 16(1), 1–25 (2006)
9. Wilensky, U., Evanston, I.L.: *NetLogo*. Center for Connected Learning and Computer Based Modeling, Northwestern University (1999)
10. Railsback, S.F.: Agent-based based Models in Ecology: Patterns and Alternative Theories of Adaptive Behaviour. In: *Agent-Based Computational Modelling*, pp. 139–152. Physica-Verlag (2006)
11. Horstmann, T., Bentley, R.: Distributed authoring on the Web with the BSCW shared workspace system. *StandardView* 5(1), 9–16 (1997)
12. Becker, R., Becker, B., Knotte, M., KreiBlemeyer, I.: *Manual eGroupware 1.4*. Creative Commons (2007)
13. Yang, X., Allan, R.: Web-Based Virtual Research Environments (VRE): Support Collaboration in e-Science. In: *WI-IATW 2006: Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*, pp. 184–187. IEEE Computer Society, Los Alamitos (2006)
14. Severance, C., Hardin, J., Golden, G., Crouchley, R., Fish, A., Finholt, T., Kirschner, B., Eng, J., Allan, R.: Using the Sakai collaborative toolkit in e-Research applications. *Concurrency and Computation: Practice and Experience* 19(12), 1643–1652 (2007)
15. Reenskaug, T.: The Model-View-Controller (MVC) Its Past and Present. *JavaZONE Conference*, Oslo (2003)

Methodological Steps and Issues When Deriving Individual Based-Models from Equation-Based Models: A Case Study in Population Dynamics

Ngoc Doanh Nguyen^{1,2}, Alexis Drogoul^{1,2}, and Pierre Auger¹

¹ IRD, GEODES UR 079,

32 avenue Henri Varagnat, 93143 Bondy Cedex, France

² MSI, IFI, Hanoi, Vietnam

42 Ta Quang Buu street, Hai Ba Trung District, Hanoi, Vietnam

doanhbondy@gmail.com, alexis.drogoul@gmail.com,

pierre.auger@ird.bondy.fr

Abstract. An important question in the simulation of complex systems concerns the emergence of global behaviours and how to model them. Individual-based models (IBM), on one hand, are designed precisely for exploring emergent phenomena, but they must be simulated (sometimes extensively) in order to detect the behaviours that could emerge at the global level. Moreover, there are no “theories of IBM” that would allow modellers to make predictions about the long-term emerging behaviours they can observe. On the other hand, equation-based models (EBM), while not exploring the same causes of emergence, represent a useful tool for making predictions about global emerging behaviours of a system, especially in the long term. In this paper, we will explore the methodological issues that arise when attempting to derive an IBM from an existing EBM model in population dynamics, dedicated to exploring the dynamics of two competing populations in a “two-patch” environment.

Keywords: Individual-based models, Equation-based models, Population dynamics, Agent-based simulation, Complex systems.

1 Introduction

In the fields of physics, ecology, society and economics, two widely accepted modeling approaches coexist: equation-based models (EBM) and individual-based models (IBM). In ecology, which will constitute our reference domain in this paper, the essence of the individual-based approach is to reproduce the properties of ecological systems by derivation of the (modeled) properties of the individuals constituting these systems. See, for example in [17] where the author explains why ecological models should be based on individuals; and how to build individual-based models in ecology. Equation-based models (normally a set of differential/difference equations) have an even longer history in ecology ([2], [5]).

Each of these two approaches has its own strengths and weaknesses (see [3], [4]) and, when it comes to modeling a complex system, the approach used depends on the

purpose of the model. EBM (for example, compartment models) operate on global laws generally, defined by the equations that apply to all members of the compartments. For example, in early ecological models, the state variables (compartments) in the models of population dynamics were often chosen as the total population densities and the model was a set of nonlinear, coupled, ordinary differential equations or discrete equations ([2], [5]). In such classical models, the individuals are assumed to be homogenous and well mixed: they are all treated as identical. The benefit of these simple EBM is that they can be handled analytically. However, given this previous assumption, they cannot be very realistic. By contrast, IBM can readily include heterogeneity in the attributes and behaviors of the individuals, in the network of their interactions, and even in the dynamics of the environment. Individual-based models (and especially agent-based models) are much more realistic with respect to the data available in the field, but they are arguably more difficult to calibrate (as the number of their parameters is usually larger) and the modelers can only rely on repeated simulations to interpret their dynamics.

In recent years, some “more realistic” EBM have been developed, which are aimed at taking different categories of individuals into account. These EBM involve a large number of variables and are in general difficult to handle analytically. However, in most cases, it is possible to consider different time scales: a fast one for processes operating at the individual level; and a slow one at the levels of the population and the community. It is then possible to use the “variables aggregation” method, deriving a reduced model which governs a few global variables at the slow time scale while taking into account all the processes going on at the individual level [9]. In most cases, this “aggregated” model is simple enough to allow for a complete study. Furthermore, this reduced model can be used to make predictions about the dynamics of the initial complete model. Interesting examples of EBM in which different categories of individuals were considered, and for which aggregation methods were used successfully to proceed to analysis of the EBM can be found in both population dynamics [1] and in prey-predator models ([7], [9], [10]).

The central question of all these modelling techniques concerns the emergence of global behaviours: how to handle and understand these behaviours using a model? In IBM, multiple repeated simulations and a thorough exploration of the parameters’ space are the key to finding behaviours emerging at the global level. Yet, even when they have been observed, there is nothing in the model allowing one to make predictions about emerging behaviours that might be observed for different datasets. There are no general theories linking the knowledge of the individuals’ rules to the emergence of a global property. On the contrary, particularly in the EBM mentioned above, the “aggregated” model is an invaluable tool to make predictions about possible emerging behaviours of the system at a global level and in the long-term. Both IBM and EBM can be seen as different modelling approaches addressing the same problem of emergence of global properties using different assumptions and different knowledge about the real system. Each approach answers different, yet complementary, questions. While IBM can help in exploring and explaining the local causes of global phenomena, EBM are useful for predicting their long-term evolution without having to explore them through simulated experiments. Therefore, it is of primary importance, in many domains, to be able to couple these two techniques when studying complex systems. This leads to some difficult questions: how do we compare the

emerging properties obtained by IBM and EBM for related cases? How can we make sure that both models obtain the same results? For example, will the EBM will remain useful in predicting the behavior of the IBM? Will the IBM produce compatible global dynamics? We claim, in this paper, that most of these questions actually pertain to the construction of the two types of models and can be partially answered by a careful methodological approach to the design of “compatible” individual-based models given an existing equation-based one and vice-versa.

In the paper we address the first part of this issue by detailing the steps and issues involved in building individual-based models from equation-based ones. We base our proposal on a case-study in population dynamics, namely, a new type of competition model. In section 2, we begin by briefly introducing the reader to the existing literature on mathematical competition models, their outcomes and problems. We then propose a more realistic model by introducing fast density-independent migration as a possibility to overcome the previous limitations. In section 3, we address the question of how to methodologically proceed from an equation-based model to a “compatible” individual-based one. We present how we have applied this methodology to our case study in section 4, and some preliminary experimental results in section 5. Finally, conclusion and discussion are summarized in section 6.

2 Case Study

2.1 Classical Competition Model: Principle Competitive Exclusion

The Interspecific Competition Model is a pioneering work of Lotka and Volterra in the beginning of twentieth century [5]. The classical model is time-continuous, deterministic and is given by the next two equations:

$$\frac{dN_i}{dt} = r_i N_i \left(1 - \frac{N_i}{K_i} - b_{ij} \frac{N_j}{K_i} \right), \quad i, j = 1, 2, i \neq j \tag{1}$$

where $N_i(t), i = 1, 2$ are total densities of species i at time t ; $r_i, i = 1, 2$ are growth rates; $K_i, i = 1, 2$ are carrying capacities of each species. $b_{ij}, i, j = 1, 2, i \neq j$ are interspecific competitive coefficients representing the negative effect of species j on the growth of species i .

To reduce the number of parameters, it is usual to nondimensionalize the previous model by changing variables and parameters as follows:

$$u_i = \frac{N_i}{K_i}; a_{ij} = b_{ij} \frac{K_j}{K_i}, \quad i, j = 1, 2, i \neq j \tag{2}$$

Under these changes, equations (1) become:

$$\frac{du_i}{dt} = r_i u_i (1 - u_i - a_{ij} u_j), \quad i, j = 1, 2, i \neq j \tag{3}$$

Whenever intra-specific competition is stronger than inter-specific competition ($a_{ij} < 1$) there is coexistence, otherwise one of species will out-compete the other (see [5]). These predictions are massively corroborated by experience and observation. This is a well-known result, also called the “competitive exclusion principle”: to coexist, species must differ in their resource use; otherwise one of them ends up extinct. However, recent experimental data show that coexistence is even possible for two or more species that are locally impermanent ([16]). Motivated by this challenge to classical competition theory, many authors proposed models which predict the role of dispersal/migration in the dynamics ([11], [12]).

2.2 Presentation of Our Model of Competition in a Two-Patch Environment

We consider an EBM, which is introduced and detailed in [6] to present a competition model in which two species compete for a common resource in an environment divided into two patches. We assume that migration between the two patches is fast in comparison to local population growth and mortality. We note population density of species 1 on patch 1 (n_{11}), species 1 on patch 2 (n_{12}), species 2 on patch 1 (n_{21}) and species 2 on patch 2 (n_{22}). Time evolution of species densities on each patch is assumed to follow the classical Lotka-Volterra interspecific competition model that we recalled in the previous section.

The flow per unit of time of individuals of species 1 which migrate from patch 1 to patch 2 is $\bar{k}n_{11}$, and from patch 2 to patch 1, kn_{12} . Similarly, the flow of individuals of species 2 which migrate from patch 1 to patch 2 is $\bar{m}n_{21}$, and from patch 2 to patch 1, mn_{22} (k, \bar{k}, m, \bar{m} are constant and positive migration rates). Because we have assumed that migration is fast in comparison to local growth and mortality on each patch, in the EBM we introduce a small positive parameter ϵ , which represents the ratio between the two time scales. Therefore, our EBM is a set of four ordinary differential equations as follows:

$$\begin{aligned}
 \frac{dn_{11}}{d\tau} &= (kn_{12} - \bar{k}n_{11}) + \epsilon r_{11} n_{11} \left(1 - \frac{n_{11}}{K_{11}} - a_{121} \frac{n_{21}}{K_{11}} \right) \\
 \frac{dn_{12}}{d\tau} &= (\bar{k}n_{11} - kn_{12}) + \epsilon r_{12} n_{12} \left(1 - \frac{n_{12}}{K_{12}} - a_{122} \frac{n_{22}}{K_{12}} \right) \\
 \frac{dn_{21}}{d\tau} &= (mn_{22} - \bar{m}n_{21}) + \epsilon r_{21} n_{21} \left(1 - \frac{n_{21}}{K_{21}} - a_{211} \frac{n_{11}}{K_{21}} \right) \\
 \frac{dn_{22}}{d\tau} &= (\bar{m}n_{21} - mn_{22}) + \epsilon r_{22} n_{22} \left(1 - \frac{n_{22}}{K_{22}} - a_{212} \frac{n_{12}}{K_{22}} \right)
 \end{aligned}
 \tag{4}$$

This model can be analyzed using mathematical perturbation techniques, which are not presented here. The analysis of this model showed that under some conditions, if one of the two species out-competes the other one in both patches, the two species could coexist. Also, that the model shows that it is even possible that the dominating species is out-competed globally. In other words, for specific migration rates, a

species that is locally out-competed in both patches can coexist together with or even out-compete the dominating species. This counter-intuitive result shows the importance of short-term migration strategies on the issue of competition. It is a good example of a long-term emerging property predicted by a deterministic EBM and, for this reason, a good candidate for our purpose. In the next section, we explore how to build an IBM that can be related to the model presented above.

3 Methodology

There are many alternative ways for “distributing” an EBM into related IBMs. An IBM consists of (at least) a set of agents in an environment, with (at least) some individual processes and attributes attached to them (in order, for instance, to give birth, die, move, consume resources or perceive their local environment). Moreover, the environment can have its own dynamics. The process can then be completely bottom-up: by designing an IBM with the minimum set of hypotheses (minimal behaviors, few attributes) and then progressively adding individual properties in order to make similar phenomena emerge from it or to obtain comparable numerical results in some parameters configurations. It can also be top-down, by progressively distributing each part of the equation into individual parameters/behaviors of an initially “blank” IBM. Our methodological approach is closer to the second one: basically, we consider the EBM as a “central controller” of the different IBMs that can be produced and we progressively distribute its control, one step at a time, carefully examining which hypotheses and decisions to make at each step, until we don’t have any global equations left, but only individual properties.

3.1 First Step: A “Hybrid” Spatialized Model

The first step of our methodology is to build a hybrid model, that is, a model with a set of agents, situated in an environment with a certain topology, but where the agents remain entirely governed by the global equations and parameters of the EBM: they do not possess nor use individual attributes or behaviors. Building such a model allows us to examine, apart from the following ones, the question of the spatialization of the EBM, which is probably the most difficult part of the process. Choosing a space, or an environment, implies making fundamental choices that will affect the dynamics and possibilities of all the agents. Several questions are raised in this first step:

- The question of the topology of the environment
- The question of its boundaries
- The question of its nature (discrete, continuous, etc)
- The question of the perception it can offer to the agents, and so on.

Each of these choices is then experimented against the original EBM (non-spatialized) for key configurations of parameters and we rank them with respect to their adequation with its dynamics.

3.2 Following Steps

Next, one by one, we remove the IBM’s dependency upon the global parameters by distributing them in the new (or existing) agents’ attributes and behaviors (or a

combination of environmental and individual attributes and behaviors). Each of these “distributions” is once again tested and validated with respect to its adequacy within the EBM. The order in which the parameters are distributed may be important (in fact, the parameters that are chosen first will probably constrain the following possibilities). There may well be several iterations needed between the hybrid model and the more individualized ones.

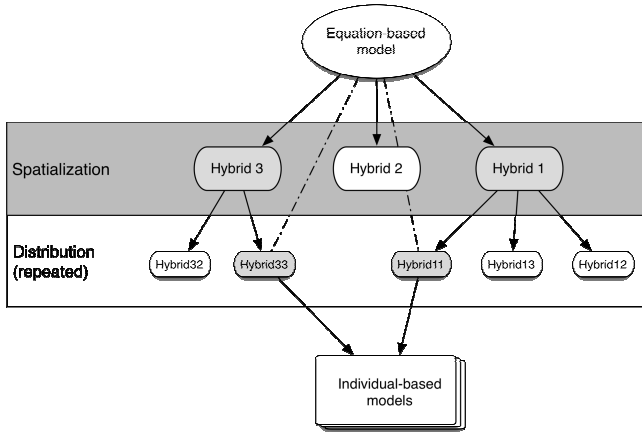


Fig. 1. The different paths between the initial EBM model and IBMs related to it. Intermediate models are hybrid IBMs in the sense that they do possess a double nature: part of their dynamics is governed by global equations. The result of this methodological process is a set of “compatible” IBMs that exhibit emergent properties similar to those described by the EBM.

Each step consists of:

- (1) Building several models related to the hypotheses we take,
- (2) Exploring their dynamics and “validate” it with respect to the EBM,
- (3) Choosing the best that fit,
- (4) Passing to another parameter.

4 Instantiation on the Case Study

4.1 Experimental Environment

In this section, we will show how we applied the methodology presented above to our case study. We used the Netlogo package to write and simulate different models. We chose Netlogo because of its multi-agent programming language and integrated modeling environment--which taken together make for easy model building. NetLogo also comes with a module called BehaviorSpace that eases the process of exploration of emergent phenomena through an automated exploration of the model’s parameters space. Finally, NetLogo allows for a side-by-side comparison of EBM and IBMs, since both representations (system dynamics and agent-based) are available for programming hybrid models.

4.2 Hybrid Model: Spatialization of the EBM and Patches

Choosing a space, or an environment, implies making fundamental choices that will affect the dynamics and possibilities of all the agents. The different possible environments available to the agents can be classified by their properties: its topology, its boundaries, its nature (discrete, continuous, etc.), the perception it can offer to the agents, and so on. We will see how we have chosen to spatialize the “two patches” environment of the EBM.

Patches

Introducing an “environment for agents” means introducing: spatial configurations, frontiers, movement, perception, density, distance, and perhaps resources available to the agents—in effect, all the things that are not detailed in the EBM and for which we must make choices. In this first step, we create a set of agents provided with basic and random spatial behaviors (i.e. they move randomly) whose demography is entirely governed by global equations controlling growth and mortality. Then, we build several models related to the way we choose to “spatialize” the two patches. We can build patches with only one boundary (with/without wraps) (see Fig.2a and Fig.2b) or with many boundaries (with/without wraps) (see Fig.2c and Fig.2d).

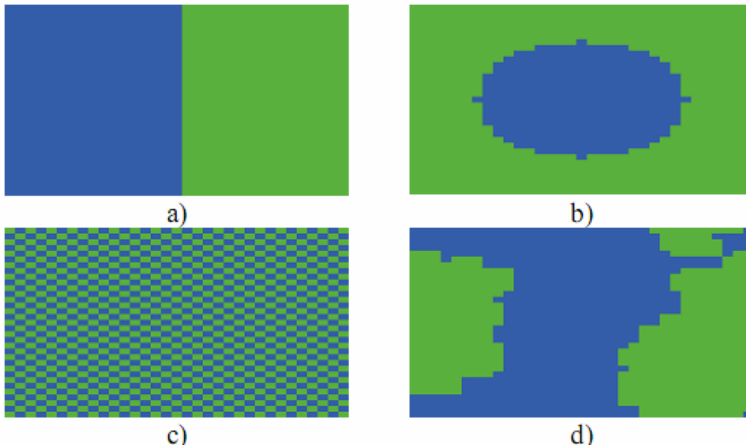


Fig. 2. a) and b): probability for migration is small; c) : probability for migration is 50%; d): probability for migration is random

These spatial configurations will strongly impact the probability and speed of the migrations between the two patches. With only one boundary, as in Fig. 2a and Fig. 2b, the probability for individuals to migrate from one patch to another is very small (they can only migrate when they are situated on the boundary). This does not really fit the EBM dynamics, since migration within it is considered “fast” compared to growth and mortality. Dividing patches, as in Fig. 2c, give a probability of 50% for individuals to migrate from one patch to another. This appears to be a better choice, as individuals have more possibilities to migrate from a patch to another. However, a set

of simulations reveals that after some time, the densities of species in both patches are equal--whatever the initial distribution--which does not fit with the results of the EBM. For example, in EBM, the proportion of densities of species 1 on patch 1 and patch 2 is equal to k_1/k_2 . In Fig2d, we choose patches randomly. This choice allows individuals to migrate easily. And the proportion of densities of species on patch 1 and patch 2 is, in general, not equal to 1. Thus, it appears to be a good candidate for the spatial configuration of the hybrid model. However, each patch in the EBM has its own carrying capacities. To translate this environment-dependent parameter into this spatial configuration, we introduce a new parameter, called “food”, possessed by each of the discrete places of the environment and that can be perceived (and consumed) by the agents. This parameter will be connected with the other processes in the following steps.

4.3 Distribution of the Movement/Migration Processes

During this first step, we chose to distribute the parameters of the movement / migration processes. In the EBM, migration rates per capita of species are constant, i.e., they represent a density-independent migration. Many questions arise when trying to remove global parameters and replace them with individual ones: for instance, is it enough just to endow each individual with a random movement/migration procedure? Are there connections between patches and movement/migration procedures? How can we interpret fast migration of the species?

Since our interest is to study the effect of fast migration on the emergent phenomenon of competitive coexistence, the movement/migration processes in the IBM should show their influence on the evolution toward coexistence. They should be fast in comparison to growth and mortality like in the EBM. To make the process faster than the other ones, we introduced a “step” to count the time between two events. The time between two events in the movement/migration process should be a lot shorter than the interval between the birth/death processes (expressing the fact that the agents have a lifetime and a reproduction rate).

The simulations of this hybrid model give us a set of distributions of individuals in the environment. We validate these distributions with respect to their adequacy within the EBM, and then we choose the best fit.

4.4 Distribution of the Birth/Death and Competition Processes

Now, we will distribute the parameters for the birth/death and competition processes. How do we convert the growth and mortality parts of the equation (4) into probabilistic rules for individual reproduction and death? How can we express competition at the individual level? Which new parameters in the IBM can correspond to these global parameters?

To convert the growth and mortality parts of the equation (4) into probabilistic rules for individual reproduction/death, we proceed as follows: Firstly, we return to the parts of the equation (4) that concern the growth and mortality of the species.

$$\begin{aligned} \frac{dn_{ii}}{d\tau} &= \varepsilon r_{ii} n_{ii} \left(1 - \frac{n_{ii}}{K_{ii}} - a_{iji} \frac{n_{ji}}{K_{ii}} \right) \\ \frac{dn_{ij}}{d\tau} &= \varepsilon r_{ij} n_{ij} \left(1 - \frac{n_{ij}}{K_{ij}} - a_{ijj} \frac{n_{jj}}{K_{ij}} \right) \end{aligned} \tag{5}$$

where $i, j = 1, 2, i \neq j$.

This continuous equation can be replaced, within the limit of very large populations, by a discrete version: where time is a discrete variable, and whose parameters should be interpreted as those in equations (5) multiplied by the time interval between generations:

$$\begin{aligned} n_{ii}(\tau + 1) &= n_{ii}(\tau) + \varepsilon r_{ii} n_{ii}(\tau) \left(1 - \frac{n_{ii}(\tau)}{K_{ii}} - a_{iji} \frac{n_{ji}(\tau)}{K_{ii}} \right) \\ n_{ij}(\tau + 1) &= n_{ij}(\tau) + \varepsilon r_{ij} n_{ij}(\tau) \left(1 - \frac{n_{ij}(\tau)}{K_{ij}} - a_{ijj} \frac{n_{jj}(\tau)}{K_{ij}} \right) \end{aligned} \tag{6}$$

Both versions (discrete and continuous) have the same critical points and stability properties. From equation (6), it is easy to see that the growth rates r_{ii}, r_{ij} represent

reproducing probabilities; extra-specific competition terms $\frac{\varepsilon r_{ii} n_{ii}^2(\tau)}{K_{ii}}, \frac{\varepsilon r_{jj} n_{jj}^2(\tau)}{K_{jj}}$

and inter-specific competition terms $\frac{\varepsilon r_{ij} n_{ij}^2(\tau)}{K_{ij}}, \frac{\varepsilon r_{ji} n_{ji}^2(\tau)}{K_{ji}}$ can be interpreted

as a death process giving death probabilities $\frac{\varepsilon r_{ii} n_{ii}(\tau)}{K_{ii}}, \frac{\varepsilon r_{jj} n_{jj}(\tau)}{K_{jj}}$ and

$\frac{\varepsilon r_{ij} n_{ij}(\tau)}{K_{ij}}, \frac{\varepsilon r_{ji} n_{ji}(\tau)}{K_{ji}}$, respectively. However, it is not easy to calculate these prob-

abilities in stochastic processes like the ones present in the IBM.

To overcome this problem, we introduce a new parameter called “energy” for the agents, whose dynamics is precisely governed by the translation of these processes at the individual level. At the beginning, each individual has its own (randomly chosen) level of energy and gains more energy over time as it eats food. They loose some energy when they move and when they meet other individuals because of competition (this corresponds, with different values, to intra-specific and inter-specific competitions). Finally, an individual dies when its energy falls to zero. Introducing this parameters means introducing a whole new set of other parameters related to consumption (how much food does an individual consume from its environment?), metabolism (how much

of its energy does it lose each step?), competition, etc. The parameters space of the IBM, already large after the first step of spatialization, becomes larger and more difficult to explore entirely.

5 Experimental Results and Comparisons

The EBM consists of two dynamical parts: at a slow time scale, a classical competition dynamic and at fast time scale, the dynamic of migration. Thus, without migration, EBM has two separatrix classical competition dynamics in two local patches, i.e., whenever intra-specific competition is stronger than inter-specific competition, there can be coexistence, otherwise one of the species out-competes the other.

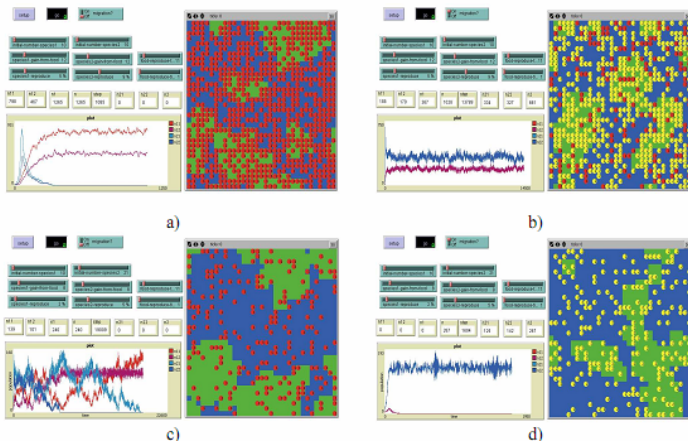


Fig. 3. a) and b): locally species1 out-compete species2 and coexistence globally; c) and d): locally species1 out-compete species2 and species2 out-compete species1 globally

In figure 3a) and 3b), we show the same result as ones in the EBM: without migration, species2 get extinct (figure 3a)) while species1 tend to the carrying capacities (figure 3b)). In certain parameter configurations, interesting results are predicted by the EBM: species1 out-competes species2 locally (i.e. in every patch) but, thanks to migration, globally, we obtain an opposite result, i.e., species1 is out-competed by species2. For the same configurations, in the IBM, we also have the same result as seen in figure 3c) and 3d).

6 Conclusion and Discussion

We have presented the first steps towards a methodology for building individual-based models that can be related to existing equation-based models. Our approach is mainly “top-down”, i.e., we progressively distribute each part of the equation into individual parameters/behaviors.

An example of this approach has been presented on a specific case-study and we have shown that, for certain configurations of parameters, the two models produce similar emerging properties (for instance, the counter-intuitive disappearance of the dominating species when migration rates are high), which can be interpreted as a first validation of our choices.

Of course, there exists an infinity of totally “artificial” IBMs that can mimic the curves obtained in an EBM and we are aware that, without any other constraints, other methodological steps could lead to similarly interesting results. However, by choosing (1) to first spatialize the model and carefully study the influence of the spatial configurations than can be chosen, then (2) to distribute, one at a time, the global parameters into individual parameters that can be related, under some conditions, to attributes existing in “real individuals” (i.e., metabolism, average lifetime, local perception of others, consumption of resources, etc.), we are confident that our methodology will enable us to address real ecological phenomena, where, for instance, only field data about the behaviors and biological attributes of individuals are available. In that case, and if we are able to build an IBM model calibrated with these field data, the work we have done will allow us to draw, much more easily than by repeated simulations, global predictions about this phenomenon (by using the related EBM).

References

1. Dubreuil, E., Auger, P., Gaillard, J.M., Khaladi, M.: Effects of aggressive behaviour on age structured population dynamics. *Ecological Modelling* 193, 777–786 (2006)
2. Edelstein-Keshet, L.: *Mathematical models in biology*. Random house, New York (1989)
3. Fahse, L., Wissel, C., Grimm, V.: Reconciling classical and individual-based approaches in theoretical population ecology: a protocol for extracting population parameters from individual-based models. *American Naturalist* 152, 838–852 (1998)
4. Georij, V.B., Goedecke, M.D., Yu, J.F., Epstein, S.M.: A hybrid epidemic model: Combining the advantages of agent-based and equation-based approaches. In: *Proceeding of the 2007 Winter Simulation Conference* (2007)
5. Murray, J.: *Mathematical Biology*. Springer, Heidelberg (1989)
6. Nguyen, N.D., Auger, P., de la Parra, R.B.: Effects of fast migrations on competitive coexistence (2008)
7. Auger, P., Pontier, D.: Fast Game Theory Coupled to Slow Population Dynamics: The case of Domestic Cat Populations. *Mathematical Biosciences* 148, 65–82 (1998)
8. Auger, P., Bravo de la Parra, R., Morand, S., Sanchez, E.: A predator-prey model with predators using hawk and dove tactics. *Mathematical Biosciences* 177, 185–200 (2002)
9. Auger, P., Bravo de la Parra, R., Poggiale, J.C., Sánchez, E., Nguyen Huu, T.: Aggregation of variables and applications to population dynamics. In: Magal, P., Ruan, S. (eds.) *Structured Population Models in Biology and Epidemiology*. Springer, Heidelberg (2008)
10. Auger, P., Kooi, B., Bravo de la Parra, R., Poggiale, J.-C.: Bifurcation Analysis of a Predator-prey Model with Predators using Hawk and Dove Tactics. *Journal of Theoretical Biology* 238, 597–607 (2006)
11. Amarasekare, N.R.M.: Spatical heterogeneity Source-Sink Dynamics and the Local coexistence of competing species. *American Naturalist* 158(6) (2001)
12. Amarasekare, P.: The role of density-dependent dispersal in source-sink dynamics. *Journal of Theoretical Biology* 226, 159–168 (2004)

13. Laubenbacher, R., Jarrach, A.S., Mortveit, H., Ravi, S.S.: A mathematical formalism for agent-based modeling, arXiv:08.01.0249v1 [cs. MA] (2007)
14. Law, R., Dieckmann, U.: Moment approximations of individual-based models (1999), <http://www.iiasa.ac.at/Admin/PUB/Documents/IR-99-043.pdf>
15. Hinckley, S., Hermann, A.J., Megrey, B.A.: Development of a spatially explicit, individual-based model of marine fish early life history. *Marine Ecology Progress Series* 139, 47–68 (1996)
16. Tilman, D.: Competition and biodiversity in spatially structured habitats. *Ecology* 75, 2–16 (1994)
17. Grimm, V., Steven, R.F.: *Individual-based Modeling and Ecology*. Princeton University Press, Princeton (2005)
18. Grim, V.: A standard protocol for describing individual-based and agents-based model. *Ecological Modelling* 198, 115–126 (2006)

Abstraction of Agent Cooperation in Agent Oriented Programming Language

Nguyen Tuan Duc and Ikuo Takeuchi

The University of Tokyo, Japan
duc@nue.ci.i.u-tokyo.ac.jp, nue@nue.org

Abstract. Collective operation is a concept of parallel programming in which many processes participate in an operation. Since collective operations are suitable for modeling the coordination of many processes, they can be used to model cooperating agents in a multiagent system. In this paper, we propose an agent oriented programming language that exploits collective operations to abstract the cooperating process of agents. We also present a method for implementing collective operations while maintaining the autonomous computational model of agent. Our experiment shows that our language and cooperation model have many advantages in developing multiagent systems...

1 Introduction

In multiagent system (MAS), agents need to exchange useful information with each other in order to reach an agreement or collaborate for achieving a goal. The process of communicating and exchanging knowledge is known as cooperation. Cooperation is a crucial requirement in MAS because without cooperation the system is simply a set of separated agents and has no ability of collaborating to reach the goal.

On the other hand, in agent oriented programming (AOP), a new programming paradigm proposed by Y. Shoham [1], agent is modeled as an autonomous, reactive and pro-active entity. Because of this autonomous computational model, the integration of autonomous agent and cooperating agent is not simple. Agents need to be autonomous, however, they also need to collaborate in order to achieve the goals.

In this paper, we propose an agent oriented programming language that supports the cooperation of agents. The language uses the concept of collective operation in parallel distributed programming to abstract the cooperating process of agents. Moreover, it maintains the autonomy of agent and provides constructs for describing agent's *mental state* (i.e., belief, desire, intention [3]). We have implemented a framework called Yaccai (Yet Another Concurrent Cooperating Agent Infrastructure) to support the execution of multiagent systems written in our language. The communication model underlying our language's execution environment ensures the autonomy of each agent while providing full support for message passing. Our experiment shows that by using collective operations,

global knowledge, an important element in multiagent systems, can be easily derived.

The rest of this paper is organized as follows. Section 2 compares our system with existing AOP languages and cooperation models. Section 3 presents our language design and language constructs for abstraction of agent cooperation using collective operations. Section 4 describes the execution model that supports the implementation of collective operations. We discuss about the application of collective operations in Section 5. Section 6 shows empirical results for evaluating the system. Finally, Section 7 discusses about future work and concludes.

2 Related Work

Research on AOP language has focused on how an autonomous agent can be described in the language, that is, how to express the mental state (the intra-agent aspects) of an agent efficiently and easily using constructs provided by the language [1][4][2]. However, existing agent oriented programming languages do not concentrate on the communication model of agents and do not pay enough attention to abstraction of agent cooperation despite the fact that cooperation is a very important issue in MAS.

Cooperation models such as joint-intentions model [5] or teamwork [7] allow the description of global goal for the entire multiagent system and coordination scheme is automatically derived from the team's goal. But it is difficult to ensure the autonomy of each agent because all agents have the same goal and mental state.

Michael Schumacher proposed a model for inter-agent coordination, called ECM [8] and a programming language to specify agent hierarchy. Agents participate in many agent societies, called "blops". Each blop is a group of agents, in which agents can easily communicate with each other and even broadcast messages when they want. However, ECM and its languages do not support the description of mental state of agent, agent itself needed to be specified by another programming language.

Our system combines the advantages of agent oriented programming languages and the coordination models mentioned above. The system ensures the autonomy of agents and provides constructs for description of cooperation, communication between agents.

3 Language Design and Abstraction of Agent Cooperation

3.1 Constructs for Modeling Mental State and Reasoning Cycle

Our language is agent oriented because it supports the description of mental state of agents and automatically generates reasoning cycle (the cycle of sense - reasoning - act). The language provides constructs to define classes and agent

classes like in normal object oriented languages. Each agent has its own independent integrated belief-base to avoid the overhead of synchronizing common belief-base and ensure the autonomy of the agent. Belief-base query/update operations are integrated in the language as language constructs that are similar to LINQ [9].

```

1  agentclass HelloAgent {
2    public m_comm;
3    public function HelloAgent() {
4      m_comm = Environment.GetCommunicator(
5        "World", MsgListener );
6    }
7    public plan MsgListener( msg ) {
8      id = -1;
9      match msg.Value with {
10     "Hello from", @ {id}, "at", @ {addr} -> {
11       belief fact new {rank=id, host=addr};
12     }
13   }
14 }
15 public plan act() {
16   myRank = Environment.GetRank();
17   m_comm.Bcast( "Hello from " + myRank +
18     " at " + Environment.Hostname() );
19 }
20 }
21
22 class SimpleMAS {
23   public static function Main() {
24     a1 = create HelloAgent() at "localhost";
25     a2 = create HelloAgent() at "somehost.com";
26   }
27 }

```

Fig. 1. A simple multiagent system definition

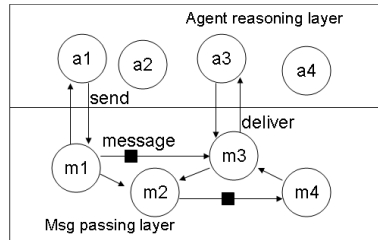


Fig. 2. Communication model

Reasoning cycle of agent is automatically realized when agent program defines a special plan with name “act”. Fig. 1 shows an example of a multiagent system definition in our language. Once the agent is created, the constructor will be called and then the plan “act” will automatically be executed. Each agent in the example simply broadcasts a “Hello” message and its identifier (rank) to others. When an agent received “Hello” message, it stores the host address of the sender into belief-base. The main program (the SimpleMAS class) uses create statement to create agents at desired host. Complete grammar of the language is available at the Yaccai’s homepage [1].

3.2 Abstraction of Agent Cooperation

We present a new approach to model the cooperating process of agents, that is, using collective operations to abstract cooperation. Collective operations are operations that involve in many processes and data of these processes of execution. For instance, broadcasting a message to all agents in a group is a collective

¹ <http://www.nue.ci.i.u-tokyo.ac.jp/%7Educ/yaccai>

operation because all agents are affected by the operation. MPI [6], a famous parallel distributed programming interface, defines many collective operations such as barrier, broadcast, gather, reduce, ... to support synchronization and cooperation of processes.

We use the concept of “communicator” in MPI (that is similar to “blob” in ECM [8]) to represent agent group, agent can freely participate into a group by invoking the following method:

```
comm = Environment.GetCommunicator( "comm_name", listenerPlan );
```

This method will get the communicator named “comm_name” and set “listenerPlan” as the message processing plan for this communicator. Every message comes to this agent from the communicator will be passed to `listenerPlan` to be processed.

An agent can leave from a communicator by invoking the “Leave” method on the communicator:

```
comm.Leave();
```

Messages come from this communicator will not be passed to “listenerPlan” anymore.

Agents form group of agents when they participate in the same communicator (by invoking the method `GetCommunicator` with the same communicator name). The concept of “communicator” in our language is the same as in MPI but collective operations’ syntax and semantics are different. Collective operations in MPI must be invoked in parallel by all processes that are participating in the operation. This requirement ensures the efficiency for the execution of collective operations but it causes difficulty in maintaining the autonomy of each process since it requires all processes invoke the operation at the same time. In our system, we allow collective operations to be invoked by just one agent and other agents will automatically participate in the operations. For example, an agent may invoke the following method to broadcast message to all agents in the same communicator:

```
comm.Broadcast( message );
```

Table 1 shows the list of collective operations that we support.

Table 1. Collective operations

Operation	Meaning	Example
Barrier	Synchronizing all agents in the communicator	<code>comm.Barrier("barrier_name");</code>
Broadcast	Broadcasting message to all agents in <code>comm</code> .	<code>comm.Bcast(msg);</code>
Reduce	Evaluate expression at each agent and apply operator on the result set in round-robin manner	<code>comm.Reduce(operator, expression);</code>
Gather	Evaluate expression at each agent then gather the results to an agent	<code>comm.Gather(expression);</code>
Scatter	Scatter the array of messages to all agents in <code>comm</code>	<code>comm.Scatter(array of msg);</code>

Collective operations allow agent to effectively cooperate with other agents in the same communicator. It makes the process of deriving global information easier. For example, an agent can get the sum of ID of all agents in the system by invoking a reduce operation: “comm.Reduce(SUM, belief query ID);”. The expression “belief query ID” is evaluated at each agent and the results are summed up by the operator SUM. Section 5 shows more about application of collective operations. It is important to note that, collective operations abstract the cooperation of agents, the abstraction simplifies the description of cooperating process.

4 Communication Model and Execution Environment

The execution of collective operation is not simple because it involves in all agents while the operation is invoked by just one agent. To cope with this problem, we use a new execution and communication model for agents as shown in Fig. 2. Each agent is divided into two layers: the agent reasoning layer and the message passing layer. Reasoning layer contains user’s code for the agent program while message passing layer contains code of the execution environment (the system provides primitives for message passing and collective operations). The former contains exactly one thread while the later may contain several threads of execution (each agent is mapped to a process, possibly in a remote host). When the reasoning layer’s code invokes *send* or *broadcast* method of communicator, the message will be passed to the message passing layer of the same agent first, and it is actually sent to destination in this layer. The message processing plan is responsible for reactive reasoning while the main plan is place where pro-active reasoning code could be described. By this way, programmers can easily model autonomous agent with pro-active reasoning and reactive reasoning capabilities.

The separation of agent’s reasoning code and message passing code supports the implementation of collective operations with different semantics from semantics in normal parallel distributed programming: collective operations can be invoked by just one agent, not all agents in parallel because the message passing layer does the job of passing messages independently from agent’s reasoning code.

5 Application of Collective Operations

Global knowledge is knowledge that involves in entire multiagent systems, for instance, the minimum value of a particular property of agents. Data that is distributed across many agents may be considered as global knowledge because gathering of the data involves in many agents. These kinds of knowledge can be easily obtained by using collective operations.

For example, a Vacuum Cleaner agent can know how many agents are in idle state by invoking the following reduce operation:

```
comm.Reduce( Sum, (belief query idle)[0] );
```

where Sum is an operator of 2 operands which returns the sum of these operands. The belief-base query expression is evaluated at each agent and returns a collection (contains only one element) that is 1 if the agent in idle state and 0

otherwise. The operator `Sum` is applied to the result set in a particular order (the order of the application depends on the reduce algorithm, such as tree-like or linear algorithm).

Another way to achieve the same goal is using the gather operation to gather idle state of all agents:

```
comm.Gather( (belief query idle)[0] );
```

The gather operation returns a collection contains values representing idle state of all agents in the communicator “comm”.

6 Evaluation

In this section, we provide some experiment results to evaluate our language and cooperation model.

We built a multiagent system for simulation of Vacuum Cleaner Robot problem. Each Vacuum Cleaner is represented as an agent whose capabilities are move and clean. The simulation server constructs a virtual space which is a grid of 20x30 cells; each cell contains zero or more units of dirt. Agent can only move up, down, left or right (one step for each cycle) in the virtual space and it receives information about the current position (e.g., number of units of dirt in the cell) from the server. The agent can only clean a unit of dirt in each cycle by sending a “clean” command. The Vacuum Cleaner agent team is written in our language (complete source code for the agent can be viewed at the Yacciai’s homepage²). A game lasts for 1000 cycles; at each cycle, the server reports the score of the game by the following formula:

$$Score = \frac{Total\ units\ of\ dirt\ cleaned}{Total\ units\ of\ dirt} \times 100 \quad (1)$$

In the first experiment, we created a multiagent system which contains 3 simple agents: the agents do not cooperate with each other, they only send/receive commands and information from the simulation server. The agents simply scan the virtual space by moving horizontally first then go up/down one row when they could not move in horizontal direction and change the horizontal direction from left-right to right-left and vice versa.

In the second experiment, we created a multiagent system which contains 3 agents that cooperate using broadcast operation. The agents use the same strategy in the first experiment to move around the virtual space. When an agent found a cell has dirt, it will broadcast the position of the cell and the units of dirt contained there to all other agents. When received message from other agents, an agent will store the information into its belief-base and determine if it should go to the cell to clean or not. When an agent successfully cleaned a position, it also broadcasts the information to another agents. An agent will

² <http://www.nue.ci.i.u-tokyo.ac.jp/%7Educ/yacciai>

Table 2. Average score of 5 times of simulation

Map	NoComm	Bcast	Reduce
map_dense	32.7(± 0)	24.28(± 0)	32.2 (± 0.3)
map_sparse	75.22(± 0)	90.91(± 0.5)	86.6 (± 0.1)

remove the dirt’s position from its belief-base when it received the clean message from other agents.

In the third experiment, we created a multiagent system which contains 3 agents that use similar cooperation scheme to the agent in the second experiment except that when an agent found dirt, it uses reduce operation to know the nearest idle agent. Then it sends the information about the cell to that agent only (not broadcast to all agents).

Table 2 shows the score for the non-communication agents, broadcast agents and reduce agents in our experiment with two scenarios: `map_dense` contains large amount of dirt that broadly distributed across many cells in the virtual space, `map_sparse` contains large amount of dirt that comparatively concentrated on a region in the virtual space. The experiment is carried on a cluster of 6 machines connected by Gigabit ethernet to guarantee that each agent is executed on a separate machine. The score is the average score of 5 times of simulation reported at the end of each simulation, the numbers after symbol \pm inside the parenthesis are standard deviations (standard deviation is very small because we do not use any random parameter).

In `map_dense`, the broadcast agents do not perform very well because they can not scan entire the space to find dirt and do too many useless moves. They seem to concentrate on a place at each time (because when received broadcast message they often go to the dirt place if they are in idle state or when they become idle, they will query the belief-base for the cell and go to clean). The reduce agents have the same performance with non-communication agents because only one agent is affected by the message when a dirt position is found. When there are many places have dirt, the agents are busy (not in idle state) and they will not react to messages from other agents immediately so the behavior of the team is similar to non-communication team. In the `map_sparse` map, the situation is different. The broadcast team has the best performance because they do not spend a lot of time in idle moves. They can concentrate on a dirt place immediately when an agent found the dirt place. The reduce team also has relatively good performance because when agents are in idle state, they will react to messages from other agents immediately so the behavior is similar to broadcast team. Agents team without cooperation has poor performance in this situation because agents have to find dirt independently and do many useless moves to scan the virtual space.

The result confirms that complex cooperation protocols may be easily described using collective operations. In situations where cooperation becomes very important (e.g., in the `map_sparse`), collective operation is very effective because it simplifies the description of cooperation of agents. Even in situations

where cooperation is not important (e.g., in the `map_dense`), the agent team uses collective operation may also achieve good performance if it uses appropriate co-operation scheme to reduce the risk of “over-cooperating” (i.e., too concentrated on a region).

7 Conclusion and Future Work

We have presented a new agent oriented programming language in which agent co-operation is highly abstracted by using collective operations. The communication model underlying our system ensures the autonomy of agents while providing full support for message passing and coordination. The system is therefore suitable for developing multiagent system, in which agents are autonomously, pro-actively and reactively taking actions while cooperating with others to achieve the goal. We also discussed about the application of our model in deriving global knowledge and shows the experiment result that confirms the effectiveness of our cooperation model. We are going to implement several real-world multiagent system benchmarks (such as RobocupSoccer or RobocupRescue agent team) to investigate the effectiveness of the language and the communication model.

References

1. Shoham, Y.: Agent oriented programming. *Artificial Intelligence* 60(1), 51–92 (1993)
2. Hindriks, K.V., et al.: Architecture for Agent Programming Languages. In: Proc. of the 14th European Conference on Artificial Intelligence (ECAI 2000) (2000)
3. Rao, A., Georgeff, M.: Modeling rational agents within a BDI architecture. In: Proc. of the Intl. Conf. on Knowledge Representation and Reasoning KR 1991 (1991)
4. Rao, A.: AgentSpeak(L): BDI Agents speak out in a logical computable language. In: Proc. of the 7th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (1996)
5. Jennings, N.R.: Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* 75(2), 195–240 (1995)
6. The Message Passing Interface, <http://www-unix.mcs.anl.gov/mpi/>
7. Pynadath, D., et al.: Toward team-oriented programming. In: Jennings, N.R. (ed.) ATAL 1999. LNCS, vol. 1757, pp. 233–247. Springer, Heidelberg (2000)
8. Schumacher, M.: Objective coordination in multi-agent system engineering. LNCS, vol. 2039. Springer, Heidelberg (2001)
9. The LINQ project, <http://msdn.microsoft.com/en-us/netframework/aa904594.aspx>

Knowledge Assessment: A Modal Logic Approach

Vineet Padmanabhan^{1,*}, Guido Governatori^{2,**}, and Subhasis Thakur²

¹ Department of Computer & Information Sciences, University of Hyderabad, India
vineetcs@uohyd.ernet.in

² School of ITEE, The University of Queensland, Brisbane, Australia
{guido,subhasis}@itee.uq.edu.au

Abstract. The *possible worlds semantics* is a fruitful approach used in Artificial Intelligence (AI) for both *modelling* as well as *reasoning* about knowledge in agent systems via modal logics. In this work our main idea is not to model/reason about knowledge but to provide a theoretical framework for *knowledge assessment* (KA) with the help of *Monatague-Scott* (MS) semantics of modal logic. In KA questions *asked* and answers *collected* are the central elements and knowledge notions will be defined from these (i.e., possible states of knowledge of subjects in a population with respect to a field of information).

Keywords: Modal & Epistemic Logics for Question Answering Systems, Question processing, Interpretation models.

1 Introduction

Modelling and reasoning about knowledge in agent systems is an active research area within the AI community [1,2]. It is often the case that the logical tool used to represent and reason about knowledge is that of modal logic [3] with the underlying *possible worlds* [3] model. There is also an *interpreted system* (IS) model which aims to give a computational flavour to S5 in terms of the states of computer processes [4,1] and this in turn makes it more suitable in one of the major application areas of knowledge reasoning namely Multi-Agent Systems (MAS). Recent works show that the IS Model can also be used for the specification of cognitive attitudes other than knowledge like *belief*, *desire* and *intention* (BDI) so that techniques like symbolic model checking can be used to verify the different agent properties inherent in the specification [5]. In this paper we deviate from the works above in the sense that our main idea is not to model/reason about knowledge but to provide a framework for *knowledge assessment* using some tools and techniques in modal logic.

To make the idea of knowledge assessment precise consider the list of questions given in Table 1. It is common practice that for assessing a student's knowledge in elementary mathematics question formats as in Table 1 is presented and is followed by a written examination. Thereafter the students answers are collected and finally the examiner returns an appreciation which usually boils down to a single number or percentage.

* Corresponding author.

** Supported by the Australian Research Council under the Discovery Project No. DP0558854.

¹ The modal logic KT45 (also called S5) is usually used to reason about knowledge.

Table 1. An Excerpt of a test in Mathematics

a	$2 \times 378 =$????
b	$322 \div 7 =$????
c	$14.7 \times 100 =$????
d	$6442 \div 16 =$????
e	$58.7 \times 0.94 =$????

As pointed out in [6] such a testing procedure provides limited information because provided that a student gives correct answers to questions a, c and e it only shows a numerical appreciation (60 percent) of his/her work. What it hides is the information related to the student’s knowledge/mastery in performing multiplication and deficiency in division operation. Moreover, the responses (answers) also indicate that there is some dependency among the questions. For instance, question e (a multidigit multiplication) in table 1 relies on elementary multiplication tested in question a. Consequently from a correct answer to question e we should infer a correct answer to question a. Obtaining and exploiting the most precise information from an assesment procedure is particularly needed in programmed courses as it reveals the weakness as well as strong points of the student’s preparation and hence advices for further study can be inferred. Similarly any computer assisted instruction system should entail a module for uncovering the user’s knowledge. We take motivation for this work from the *knowledge structure (KS)* theory as outlined in [6,7]. Knowledge structure theory presupposes that the *knowledge* of an individual in a particular domain of knowledge can be operationalised as the solving behaviour of that individual on a domain specific set X of *problems*. If the solution result for each problem is binarily coded by true/false, then the *knowledge state* of an individual in the given field of knowledge can be formally described as the subset of problems from X he/she is capable of solving. To tackle the problem of solution dependencies that can exist between problems of a certain field of knowledge KS theory employs the concept of a *surmise system*. The idea is to associate each problem $x \in X$ with a family of subsets of X called clauses, with the interpretation that, if a person is capable of solving x then he/she is capable of solving all problems in at-least one of these elements.

In this work we describe a theoretical framework based on the possible worlds model to capture the main ingredients of KS theory as mentioned above which in turn can be used for Knowledge Assessment. Since our main aim is with respect to the *assessment* of knowledge we need to have a definition of knowledge that can fit in with this intuition. Hence, instead of defining knowledge as truth in all possible worlds, which is the common interpretation given for knowledge models based on possible worlds semantics, an agent’s knowledge is *explicitly* described at a state/(in our case with respect to a question q) by a set of sets of states (set of prerequisites for the question q). In other words, our possible worlds framework for knowledge assessment is based on *Montague-Scott (MS)* semantics rather than the usual Kripke semantics.

In the coming sections we briefly discuss the knowledge structure theory along with surmise systems and outline the technical apparatus of MS-structures. Then we show how MS-structures can be used as a tool for Knowledge Assessment and conclude the paper with a discussion.

2 Knowledge Structures, Surmise Systems and MS-Models

As mentioned in the previous section a knowledge structure consists of a finite set \mathbb{Q} together with a collection \mathcal{K} of subsets of \mathbb{Q} wherein the elements of \mathbb{Q} are the *questions* and the members of \mathcal{K} are the *knowledge states*. For example, assume that the set of questions in Table (I) is given for a test. Now, any student who took the test is characterised by the subset of questions he/she correctly answered and this subset constitutes his/her knowledge state. So for instance, we can have $\mathcal{K}_1 = \{a, b, c\}$, $\mathcal{K}_2 = \{d\}$, $\mathcal{K}_3 = \emptyset$ representing respectively the knowledge states corresponding to three students. What we can infer from the knowledge states is that the first student gave correct responses to questions a, b and c whereas the last student to none at all. Similarly one can come up with a collection \mathcal{K} of knowledge states representing all possible knowledge states by observing a population of students as given in (II).

$$\mathcal{K} = \{\emptyset, \{a\}, \{d\}, \{a, b, c\}, \{a, d, e\}, \{b, c, d, e\}, \{a, b, c, d, e\}\} \tag{1}$$

It should be noted that not any subset of \mathbb{Q} needs to be a knowledge state as solution dependencies could exist among the members of the set \mathbb{Q} . Therefore \mathcal{K} comprises of all those subsets of \mathbb{Q} which constitutes the set of all *empirically expectable* solution patterns. Also, from (II) it can be seen that questions b and c belong exactly to the same knowledge states, i.e., $\{a, b, c\}$ and $\{b, c, d, e\}$. Hence as mentioned in (III) one can say that b and c *define* the same notion. But this is not the case for questions b and e because they are distinguished by the knowledge state $\{a, b, c\}$ and this means b and e test different skills. As pointed out above solution dependencies can exist between problems of a certain field of knowledge. In our case question e in Table I relies on question a and hence from a correct response to question e we should infer a correct response to a, i.e., we say that we surmise mastery of question a from mastery of question e. In general *we want to infer from the knowledge of one question the complete knowledge of at least one set of questions among some list of sets*. We call these sets the clauses for the original question q. For example let \mathbb{Q} denote the set of questions in table I and v a mapping that associates to any element q in \mathbb{Q} a non-empty collection $v(q)$ of subsets of \mathbb{Q} as given in Figure 1. Here question a has only one clause which is that of c and question b has the empty set as its only clause. What this means is that there is only one way to know question a which is through the acquisition of question c while there is no prerequisite for b.

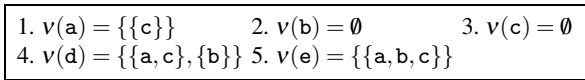


Fig. 1. Solution dependencies for the questions in Table I

Definition 1. A surmise system on a finite set \mathbb{Q} is a mapping v that associates to any element q in \mathbb{Q} a nonempty collection $v(q)$ of subsets of \mathbb{Q} and satisfies the following conditions;

1. Any clause for question q contains q
2. If $q' \in C$, with C a clause for question q , there exists some clause C' for q' satisfying $C' \subseteq C$.
3. Any two clauses for question q are incomparable in the sense that neither is included in the other.

We denote a surmise system by (\mathbb{Q}, ν) .

2.1 MS/Neighbourhood Semantics

Montague-Scott semantics, also known as Neighbourhood semantics is considered the most general kind of possible worlds semantics in the sense that it is compatible with retaining the classical truth-table semantics for the truth-functional operators. In this section we outline the main ingredients of neighbourhood semantics needed to develop a framework for knowledge assessment.

Definition 2. A neighbourhood model is a structure

$$\mathcal{M} = \langle \mathbb{W}, \pi, \nu \rangle$$

where \mathbb{W} is a set of worlds and $\pi(w)$ is a truth assignment to the primitive propositions for each state $w \in \mathbb{W}$. Intuitively $\pi(p) = \{w_1, w_2\}$ represents the fact that p is true at w_1, w_2 and false at $\mathbb{W} \setminus \{w_1, w_2\}$. $\nu(w)$ is a mapping from \mathbb{W} to sets of subsets of \mathbb{W} , i.e., $\nu : \mathbb{W} \rightarrow \wp(\wp(\mathbb{W}))$. $\langle \mathbb{W}, \nu \rangle$ is called a neighbourhood frame.

The basic idea of this definition is that each world w of \mathbb{W} has associated with it a set $\nu(w)$ of propositions that are necessary at w . Since a proposition in possible worlds semantics is a subset of \mathbb{W} ² the set of propositions necessary at w , $\nu(w)$, is a set of subsets of w . There are no assumptions about ν except that it is a function from $\mathbb{W} \rightarrow \wp(\wp(\mathbb{W}))$ and $\nu(w)$ may be any set of propositions including the empty set. When interpreted in terms of knowledge in agent-systems the members of $\nu(w)$ can be considered as the propositions an agent knows. We will talk more about this knowledge interpretation in the next section. In order to state the truth conditions of a neighbourhood model we need to take care of the definition of a truth set.

Definition 3. The truth set, $\|A\|^{\mathcal{M}}$, of the formula A in the model \mathcal{M} is the set of worlds in \mathcal{M} at which A is true; formally

$$\|A\|^{\mathcal{M}} = \{w \text{ in } \mathcal{M} : \mathcal{M}, w \models A\}$$

Definition 4. (Truth Conditions) Let w be a world in a model $\mathcal{M} = \langle \mathbb{W}, \pi, \nu \rangle$.

- $\mathcal{M}, w \models \Box A \Leftrightarrow \|A\|^{\mathcal{M}} \in \nu(w)$
- $\mathcal{M}, w \models \Diamond A \Leftrightarrow (\mathbb{W} - \|A\|^{\mathcal{M}}) \notin \nu(w)$

Example 1. Let $\mathbb{W} = \{a, b, c\}$, $\pi(p) = \{a, b\}$, $\pi(q) = \{b, c\}$ and $\nu(a) = \{\{b\}, \{a, c\}\}$, $\nu(b) = \{\{a, c\}, \{a\}, \{a, b\}\}$ and $\nu(c) = \{\emptyset, \{a\}, \{b, c\}\}$ be a neighbourhood model \mathcal{M} according to Definition 2. Then some of the formulae that are satisfied by \mathcal{M} are

² In possible worlds semantics (any kind) a proposition is identified with a set of possible worlds.

$$\begin{aligned}
 \mathcal{M}, b &\models \Box p && (\text{since } \| p \|^\mathcal{M} = \{a, b\} \in v(b)) \\
 \mathcal{M}, b &\models \Diamond p && (\text{since } \{a, b, c\} - \| p \|^\mathcal{M} = \{a, b, c\} - \{a, b\} = c \notin v(b)) \\
 \mathcal{M}, c &\models \Box \Diamond p && (\text{since } \| \Diamond p \|^\mathcal{M} = \{b, c\} \in v(c)) \\
 \mathcal{M}, a &\models \Box \Box p && (\text{since } \| \Box p \|^\mathcal{M} = \{b\} \in v(a)) \\
 \mathcal{M}, c &\models \Box \perp && (\text{since } \| \perp \|^\mathcal{M} = \emptyset \in v(c)) \\
 \mathcal{M}, a &\models \Box(p \wedge q) && (\text{since } \| p \wedge q \|^\mathcal{M} = \{b\} \in v(a)) \\
 \mathcal{M}, a &\not\models \Box p && (\text{since } \| p \|^\mathcal{M} = \{a, b\} \notin v(a))
 \end{aligned}$$

The last two items in the above list needs special mention. Note that $\mathcal{M}, a \models \Box(p \wedge q)$ but $\mathcal{M}, a \not\models \Box p$. In the case of a relational structure if we fix the valuations of p and q it is not possible to show that $\Box(p \wedge q)$ is true at a but $\Box p$ is false at a . The reason is that $\Box p$ is false at a forces a to have an accessible world in which p is false. There is only one such world (c) where p is false. However, if c is accessible from a , then $\Box(p \wedge q)$ will no longer be true at a (since if p is false at c then so is $p \wedge q$). The above example shows that the axiom $\Box(\psi \wedge \phi) \rightarrow \Box\psi \wedge \Box\phi$ is not valid in the case of neighbourhood frames. In the next section we will demonstrate why such axioms need to be avoided in th case of knowledge assessment.

3 Assessing Knowledge

In this section we show how to use the technical apparatus of Neighbourhood models as outlined above for knowledge assessment. We write the modal connectives as \mathbf{K} to emphasise the knowledge aspect. Initially we do not want to bind \mathbf{K} with any properties but just as a replica of the modal operators.

Consider a neighbourhood model $\mathcal{M} = \langle \mathbb{W}, \pi, v \rangle$ where $\mathbb{W} = \{a, b, c, d, e\}$ be the set of questions as given in Table 1 and v be as in Figure. 1 Let π be given as follows

$$\pi(*) = \{a, c\}, \pi(\div) = \{b\}$$

By $\pi(*) = \{a, c\}$ we mean that multiplication is true/holds for questions a and c . For question e this need not be the case because to solve e one needs the knowledge of both multiplication and division. Similar argument holds in the case of $\pi(\div) = \{b\}$. Now we can say that a model \mathcal{M} and question q satisfies the knowledge of multiplication if and only if the truth set of multiplication is in the list of sets related to question q . Formally

$$\mathcal{M}, q \models \mathbf{K}(*) \Leftrightarrow \| * \| \in v(q) \quad (2)$$

To give an example if we substitute question a from Table 1 in place of q we get

$$\mathcal{M}, a \not\models \mathbf{K}(*) \quad (\text{since } \| * \|^\mathcal{M} \notin v(a)) \quad (3)$$

because $\| * \| = \{a, c\}$ and $\{a, c\} \notin v(a)$. From a knowledge assessment perspective (3) has much to offer. For instance, suppose that we have a collection \mathcal{K} of knowledge states as given in (1) in Section 2. where we have a set $\{a\}$. Then (3) shows the incomplete knowledge of a student with respect to multiplication. In other words (3) helps in assessing a student's knowledge in multiplication with respect to (from the viewpoint of) the answer set provided by him/her. In this case we can assess that a correct response

to question a is not enough for a student to solve (have complete knowledge of) other questions related to multiplication. In a similar manner from (3) we can also reason about a student’s lack of knowledge in *division* because

$$\mathcal{M}, a \not\models \mathbf{K}(\div) \text{ (since } \parallel \div \parallel^{\mathcal{M}} \notin v(a)\text{)}.$$

It should be kept in mind that it is possible to make the model \mathcal{M} satisfy certain conditions so as to fit in with the notion of a surmise system as outlined in the previous section. For instance, the first item of Definition 1 generalises the reflexivity condition for a relation and we will show later on how to give such conditions for a neighbourhood model \mathcal{M} . Now, let us take d and repeat the same process. This time we can see that

$$\mathcal{M}, d \models \mathbf{K}(\ast) \text{ (since } \parallel \ast \parallel^{\mathcal{M}}, \text{ i.e., } \{a, c\} \in v(d)\text{)} \tag{4}$$

holds which tells us that a student who has provided the answer set d *knows* or have mastered multiplication. From (4) we can also infer

$$\mathcal{M}, d \models \mathbf{K}(\div) \text{ (since } \parallel \div \parallel^{\mathcal{M}}, \text{ i.e., } \{b\} \in v(d)\text{)} \tag{5}$$

which shows that a student who has provided the answer set d *knows* division. At the same time from (4) and (5) we get

$$\mathcal{M}, d \not\models \mathbf{K}(\ast \wedge \div) \text{ (} \parallel \ast \wedge \div \parallel^{\mathcal{M}}, \text{ i.e., } \{a, b, c\} \notin v(d)\text{)} \tag{6}$$

which tells us that from answer set d one cannot assess the knowledge of both multiplication and division. For instance, from Figure 1 it can be seen that question d can be mastered along two different approaches, one implying the mastery of the sole question b, the other requiring the mastering of questions a and c. In other words, according to our model, for a student to solve question d he/she needs to know multiplication or division and not both. (6) shows exactly this and more in the sense that it avoids the problem of *logical omniscience* (LO)³ [3][8] which plague knowledge models based on possible worlds. Now let us consider e;

$$\mathcal{M}, e \models \mathbf{K}(\ast \wedge \div) \text{ (} \parallel \ast \wedge \div \parallel^{\mathcal{M}}, \text{ i.e., } \{a, b, c\} \in v(e)\text{)} \tag{7}$$

(7) shows the mastery/knowledge of a student in multiplication and division with respect to question e or in other words a student who has provided the answer set e knows both multiplication and division. There are two main reasons for having such an assessment procedure; 1) It is usually the case that in an oral examination teachers strongly reduce the number of questions by making inferences from the collected answers and 2) because of 1 they can specifically select the next question. These two features also show the superior efficiency of oral testing over written testing. Any good automated procedure should encompass these features and exploit them to minimise the test duration. Our aim in this paper is to give a theoretical model based on modal logic to account for the above mentioned features. Of course there are other models (probabilistic models) that can account for such an assessment procedure but the main idea here is to show the usability of modal logic as a tool for knowledge assessment.

³ LO usually refers to a family of related *closure* conditions. In the case of (6) we avoid *closure under conjunction*, i.e., the condition that if an agent *i* knows both φ and ψ , then agent *i* knows $\varphi \wedge \psi$.

3.1 Models Satisfying Certain Conditions

So far we have been trying to build a framework based on modal logic for knowledge assessment so as to decide what formulas should be valid for the *knowledge* reading of \Box (i.e. when we interpret \Box to be a modality representing knowledge). We did not impose any constraints on the model \mathcal{M} . Since we want to relate our knowledge assessment model with that of a surmise system we need to make sure that our model satisfies conditions given in Definition 1. In this section we show how to achieve this. The following conditions can be given for items 1, 2 and 3 of Definition 1

1. $X \in v(w) \Rightarrow w \in X$ (*reflexivity condition*)
2. $X \in v(w) \Rightarrow \{w' \in W : X \in v(w')\} \in v(w)$ (*transitivity condition*)
3. $\forall X, Y \in v(w), X \neq Y \Rightarrow \exists x, y : x \in X, x \notin Y, y \in Y, y \notin X$ (*Any two clauses are incomparable*)

It should be kept in mind that given a function $v : W \rightarrow \wp(W)$ it is always possible to define a function $f : \wp(W) \rightarrow \wp(W)$ such that $f(X) = \{w : X \in v(w)\}$. In this manner we can define every function v of Definition 2 in terms of a function like f as follows;

$$w \in f(X) \Leftrightarrow X \in v(w) \quad (8)$$

Hence truth conditions for $\Box A$ in terms of f can be given as

$$\begin{aligned} \mathcal{M}, w \models \Box A &\Leftrightarrow \|A\|^{\mathcal{M}} \in v(w) \Leftrightarrow w \in f(\|A\|^{\mathcal{M}}), \text{ i.e.,} \\ &\|\Box A\|^{\mathcal{M}} = f(\|A\|^{\mathcal{M}}) \end{aligned}$$

The corresponding model conditions using (8) for reflexivity ($f(X) \subseteq X$) and transitivity ($f(X) \subseteq f(f(X))$) is much more concise and easy to use. This alternate characterisation of v -models is nothing more than a notational variant and should not be seen as a new model. A question which naturally comes to mind then is why not define conditions like reflexivity, transitivity etc. before hand on the set of questions so as to have a relational model (A *binary relation* on the set of questions so as to formalise the surmise idea). rather than constructing a surmise system as discussed in the previous sections. One reason for not adopting a relational model as pointed out in [6] is that the knowledge structure associated to a surmise relation is closed under intersection and union whereas that of a surmise system is closed under union alone. Put in other words, if two students characterised by their knowledge states K and K' meet and share what they know they will both end with the union $K \cup K'$ as their common knowledge state. In the case of intersection similar motivation doesn't exist and the only argument that could be given is that the two students would decide to retain their common knowledge, i.e., $K \cap K'$ which according to [6] is weak because cognitive development is considered to be cumulative over time. And from a modal logic point of view we can avoid the problem of LO which as pointed out earlier is not a good property to have as far as knowledge assessment is concerned. Also, in the relational model the accessibility relation must be given before defining satisfiability in a world because the satisfiability of a formula containing a modal operator is defined in terms of the accessibility relation. We can avoid this using the MS-models.

4 Discussion

We have outlined a modal logic based approach for knowledge assessment where questions asked and answers collected form the main ingredients and knowledge notions are defined from these. Our approach is different when compared to other modal logic theories of knowledge in Artificial Intelligence where modelling/reasoning about knowledge is the main focal area. The current work is in the preliminary stages and lot needs to be done. We have only outlined the syntax and semantics of our framework and have completely neglected the multi-agent aspect. What we would like to have ideally is to efficiently uncover, given a student in the population, which member of \mathcal{K} represents his/her knowledge state. From a multi-agent perspective we can think of modifying v to v_i where i represents an agent and assign the propositions he/she knows. But in the case of knowledge assessment it is not that simple because we cannot assign randomly the questions a particular agent/student knows as the assessment is done based on the questions asked and answers collected. An earlier version of this paper appeared in [9].

References

1. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning About Knowledge. MIT Press, Cambridge (1995)
2. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press, Cambridge (2000)
3. Hintikka, J.: Knowledge and Belief. Cornell University Press (1962)
4. Halpern, J.Y., Zuck, L.D.: A little knowledge goes a long way: Knowledge-based derivations and correctness proofs for a family of protocols. *J. ACM* 39(3), 449–478 (1992)
5. Su, K., Sattar, A., Wang, K., Luo, X., Governatori, G., Padmanabhan, V.: Observation-based model for bdi-agents. In: *AAAI*, pp. 190–195 (2005)
6. Doignon, J.P.: Probabilistic assessment of knowledge. In: Dietrich, A. (ed.) *Knowledge Structures*. Springer, Heidelberg (1994)
7. Albert, D., Lukas, J. (eds.): *Knowledge Spaces*. Lawrence Erlbaum Associates, Mahwah (1999)
8. Moreno, A.: Avoiding logical omniscience and perfect reasoning: A survey. *AI commun.* 11(2), 101–122 (1998)
9. Padmanabhan, V., Governatori, G., Su, K.: Knowledge assessment: A modal logic approach. In: *IJCAI Workshop on Knowledge Reasoning for Answering Questions* (January 2007)

The Design of a Self-locating Automatic-Driving Robot

Gi-Duck Park, Robert McCartney, and Jung-Jin Yang

School of Computer Science and Information Engineering
The Catholic University of Korea
School of Computer Science and Engineering
The University of Connecticut
beda.park@gmail.com, robert@enr.uconn.edu,
jungjin@catholic.ac.kr

Abstract. In order for a robot to be part of ubiquitous computing to provide a solution or service in different and distributed environment, it needs to precisely recognize the environment it is placed into and the activity of the robot is controlled by the input data through this recognition. In this paper, we present a system which allows the robot to effectively accomplish its mission by actuators citing the Mental State built through integrating the context collected and data output by various component agents of the robot. The system uses the blackboard system, serving as the central data structure in intellectual activity of a robot. Also, the feasibility of this system is demonstrated through constructing a cricket-based self-locating automatic-driving robot.

1 Introduction

In order for a robot to be part of ubiquitous computing [1] to provide a solution or service in different and distributed environment, it needs to precisely recognize [2] the environment it is placed into and the activity of the robot is controlled by the input data through this recognition. However, we have come in need of a structural method and a robot design that would make use of the recognized context and collected data through various perception device in a ubiquitous environment, which is bound to distribute, that the robot is placed in. For this reason, many research are underway of robot design [4] using the Multi Agent System [3] and robot design [6] using the Blackboard System [5]. Through this paper we will illustrate a self-locating automatic-driving robot, using the component structure of a robot that will efficiently save and make use of the data collected and output by various agents of a robot.

2 Related Research

A part of distributed artificial intelligence, the Blackboard Architecture [5] provides a solution through the cooperative work of distributed agents. The Blackboard is a component of the Blackboard Architecture that is divided into several abstract levels that are suitable for the problem. The agents [3] that communicate with others through a specific level can make a transition to an adjacent level via interaction. By this method, the collected data by the agent can be shared and the shared data in turn can be used for the transition to the aimed level.

The Cricket system [7], usually constructed on the ceiling, is composed of a beacon that transmits RF and supersonic sound signal and a receiver that receives the signals sent by the beacon. The basic strategy to locate the placement is the conventional distance measurement using supersonic, which means it is a proximity method of deciding the present location using the already stored information of the beacon location measured using the Time of Flight of the sound wave. The system used in this paper is not such proximity method but a method that pinpoints the robot's location through trigonometric survey using 3 signals received from multiple beacons installed in the ceiling.

3 SOFTWARE Configuration

As illustrated in Figure 1, the IDIS Agent acts on the bases of the Sensor component, which senses the changed environment, and the Actuator component, which changes the environment, in the Qplus Linux Kernel 2.6 operating system. The IDIS Agent can be described as an intelligent entity to achieve the goal and is designed to be able of logical parallel activity by embodying the overall frame of the Agent and actuators through process. In order to deploy the IDIS Agent in the system, the system needs an ability to add actuators which can use the agent's intelligent data structure, be reactive, and execute its intelligent conception.

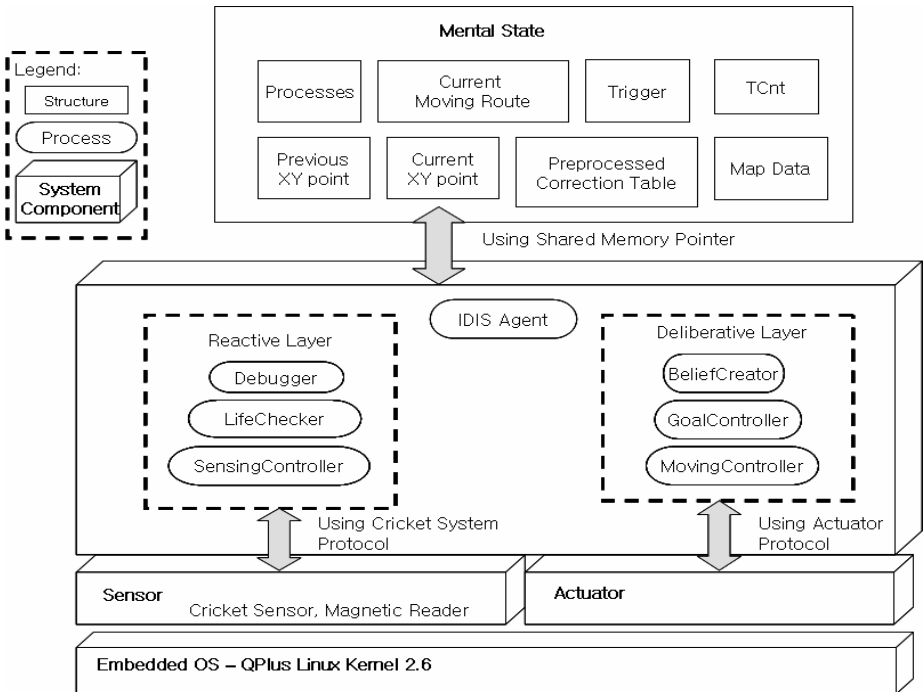


Fig. 1. The Structure of IDIS Agent System Components

IDIS Agent uses the Mental State structure indicated at the top of Figure 1, and the Mental State stores the information and knowledge of the environment recognition from the Agent's perspective. The Mental State structure is allocated at the common memory where it is accessed as reference in the processor, composing the IDIS Agent. The Mental State is defined by the combination of structures of processor information, location information, status information, environment information, and preprocessed knowledge which all compose the IDIS Agent.

As indicated in the middle of Figure 1, the actuators composing the IDIS Agent can be divided into two large categories. The first is the actuators, which correspond to the Reactive Layer that is operated regularly. These consist of actuator, which protects the agent itself and prevents it from having a negative influence on the environment by observing the environment information and status information, and another actuator, acting as the debugger the information of the Mental State. The second fall to the category of Deliberative Layer. There are three types of actuators here; ones initializing the Mental State of IDIS Agent to the belief of the agent, others fulfilling the object of the agent and thus achieving the over all goal, and still others performing the agent's movement considering the present location and status.

The Sensing Controller and Moving Controller actuators, indicated at the bottom of Figure 1, use the Cricket System protocol to interact with the Sensor component, and the Actuator protocol to interact with the Actuator component respectively. The actuator of the Sensing Controller has a regular cycle (i.e. a cycle allowed by the Cricket System) and it performs the self-protection of the agent by producing location information, used by the agent, from distinguishing the location of the IDIS Agent, using modified data through a modifying algorithm, and detecting the existence of a land mine. The Moving Controller is in charge of proper movement based on the routes that the agent should take, recorded in the Mental State.

4 Agent-Based Approach

An agent can add an actuator, which would communicate with a distributed data system, such as a multi-agent system platform like Jade or a distributed system like CORBA in order to extend and revise knowledge according to circumstances and cooperate with other agents.

The Belief Creator actuator updates the Mental State of modified data and preprocessed pathways (obstacles are applied because they are already known) by input and output of file. For example of such application, the system that we have suggested replaced the input and out put of file by TCP/IP based communicating actuator. The Debugger actuator monitors the Mental State regularly. Monitoring and extract of data based on GUI is possible if a simple application of TCP/IP is introduced. If abstract design techniques of object-oriented, rule-based, and script-based are applied, the actuators have high reusability and productivity in composing another agent. This is under the assumption that the Mental State is shared.

5 The Flow of Component

The course through which the IDIS Agent performs can be summarized by the activity diagram of Figure 2. Once the IDIS Agent is active, it uses the Belief Creator

actuator to initialize the Mental State. Afterwards, the Reactive Layer actuators are activated to modify the environment and status information in the Mental State. The process of the Sensing Controller can end up in endless wait if the Cricket-data-producing module stops while the Cricket data is being sensed. Life Checker senses the actuator in endless wait, and reboots the processor if the actuator is indeed stuck. The Sensing Controller stops the movement if the IDIS agent senses a land mine while moving, and records whether the land mine exists in the Mental State.

The IDIS Agent activates the Goal Controller actuator and the Moving Controller actuator by turn. The Goal Controller records the information and knowledge to achieve

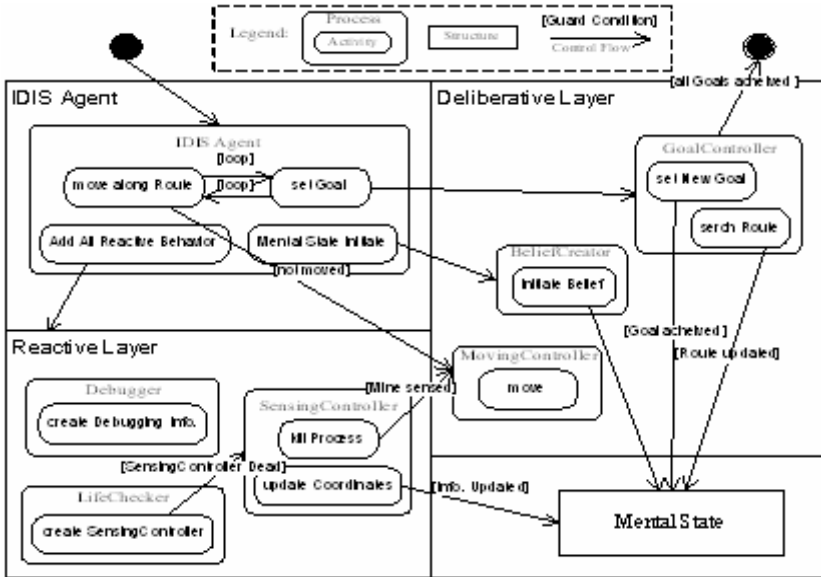


Fig. 2. IDIS Agent Activity Diagram

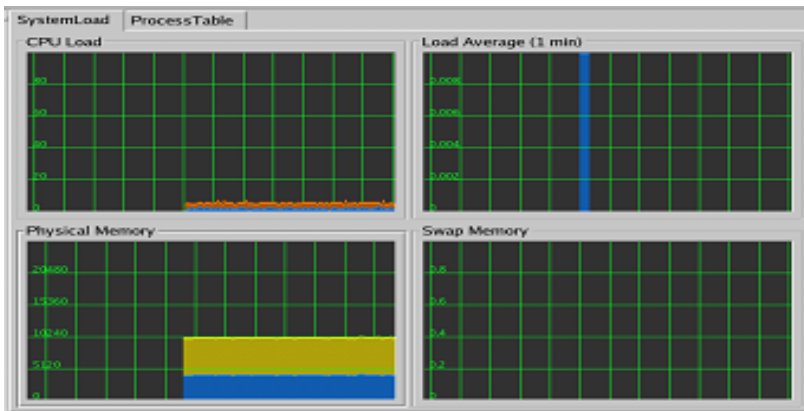


Fig. 3. Resource Measurement

each sub-goals, which have been divided to reach the overall goal, on the Mental State. It decides the pathway to achieve the sub-goal and to take additional action or set the next sub-goal depending on whether the previous one has been accomplished. Also, if a land mine has been detected, the Moving Controller sets an alternative path after a search. The Moving Controller cites the pathway and action set by the Goal Controller to reach the goal and executes necessary moves needed for the goal achievement.

6 Conclusion

The IDIS Agent leads and activates small units of actuators at the needed moment. Therefore, the resource consumption is low because all the functions of the system are not stationed as images in the memory. Also, the actuator can be improved by arranging improved process images at the activation point. Figure 3 is a target system monitor provided by Visual Esto, measuring the consumed resources until the IDIS agent retrieves and saves the target.

CPU shows low usage under 10% and the required memory maintained a consistent volume. Thus, the rest of the resources can be used for various purposes such as additional actuator and supplements. Considered methods of changing the Mental State at the point of action are to use a light DBMS to separate the data and applied code, and to produce a different form of Mental State and introduce a synchronizing process.

Acknowledgement

This research is supported by Foundation of ubiquitous computing and networking project (UCN) Project, the Ministry of Knowledge Economy(MKE) 21st Century Frontier R&D Program in Korea and a result of subproject UCN 08B3-S2-10M.

References

1. Abowd, G.D., Mynatt, E.D.: Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction* 7(1) (March 2000)
2. Chen, H., Finin, T., Joshi, A.: An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review* 18(3), 197–207 (2004)
3. Sycara, K.P.: Multiagent Systems. *Intelligent Agents AL magazine* 19(2) (Summer 1998)
4. Inocenti, B., Lopez, B., Salvi, J.: Multi-Agent System Architecture with Planning for a Mobile Robot. In: *X Conference of the Spanish Association for Artificial Intelligence, CA-EPIA* (November 2003)
5. Corkill, D.D.: Blackboard Systems. *AI Expert* 6(9), 40–47 (1991)
6. Lau, T.L., Lau, H.Y.K., Ko, A.: A Distributed Blackboard-based Control System for Modular Self-Reconfigurable Robots. *The University of Hong Kong, Department of Industrial and Manufacturing Systems Engineering* (2003)
7. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The Cricket Location-Support System. *MIT Laboratory for Computer Science* 2000 (2000)

Settling on the Group's Goals: An n-Person Argumentation Game Approach

Duy Hoang Pham^{1,2}, Subhasis Thakur¹, and Guido Governatori²

¹ The University of Queensland,
School of Information Technology and Electrical Engineering,
Brisbane, Australia

{pham, subhasis}@itee.uq.edu.au

² National ICT Australia, Queensland Research Laboratory
guido.governatori@nicta.com.au

Abstract. Argumentation games have been proved to be a robust and flexible tool to resolve conflicts among agents. An agent can propose its explanation and its goal known as a claim, which can be refuted by other agents. The situation is more complicated when there are more than two agents playing the game.

We propose a weighting mechanism for competing premises to tackle with conflicts from multiple agents in an n-person game. An agent can defend its proposal by giving a counter-argument to change the “opinion” of the majority of opposing agents. During the game, an agent can exploit the knowledge that other agents expose in order to promote and defend its main claim.

1 Introduction

In multi-agent systems, there are several situations requiring a group of agents to settle on common goals despite each agent's pursuit of individual goals which may conflict with other agents. To resolve the conflicts, an agent can argue to convince others about its pursued goal and provides evidence to defend its claim. This interaction can be modelled as an argumentation game [1, 2, 3]. In an argumentation game, an agent can propose an explanation for its goal (i.e., an argument), which can be rejected by counter-evidence from other agents. This action can be iterated until an agent either successfully argues its proposal against other agents or drops its initial claim.

The argumentation game approach offers a robust and flexible tool to resolve conflicts by evaluating the status of arguments from agents. Dung's argumentation semantics [4] is widely recognised to establish relationships (undercut, defeated, and accepted) among arguments. The key notion for a set of arguments is whether a set of arguments is self-consistent and provides the basis to derive a conclusion.

An argumentation game is more complicated when the group has more than two agents. It is not clear how to extend existing approaches to resolve conflicts from multiple agents, especially when agents have equal weight. In this case, the problem amounts to deciding which argument has precedence over competing arguments. The main idea behind our approach is the global collective preference over individual proposals, which enables an agent to identify the key arguments and premises from opposing agents in order to generate counter-arguments. These arguments cause a majority of opposing

agents to reconsider their claims, therefore, an agent has an opportunity to change “attitudes” of others.

Each of our agents is equipped with its private knowledge, background knowledge, and knowledge obtained from other agents. The background knowledge, commonly shared by the group, presents the expected behaviours of a member of the group. Any argument violating the background knowledge is not supported by the group. The knowledge about other agents, growing during the game, enables an agent to efficiently convince others about its own goal. Defeasible logic is chosen as our underlying logic for the argumentation game due to its efficiency and simplicity in representing incomplete and conflicting information. Furthermore, the logic has a powerful and flexible reasoning mechanism [5, 6] which enables our agents to flawlessly capture Dung's argumentation semantics by using two features of defeasible reasoning, namely the ambiguity propagating and ambiguity blocking.

Our paper is structured as follows. In section 2, we briefly introduce notions of defeasible logic and the construction of the argumentation semantics. Section 3 introduces our n-person argumentation game framework using defeasible logic. We present firstly the external model of agents' interaction, which describes a basic procedure for an interaction between agents. Secondly, we define the internal model, which shows how an agent can deal with individual knowledge sources to propose and defend its goal against other agents. Finally, we show the justification of arguments generated by an agent during the game w.r.t. the background knowledge of the group. Section 4 provides an overview of research works related to our approach. Section 5 concludes the paper.

2 Background

In this section, we briefly present essential notions of defeasible logic (DL) and the construction of Dung's argumentation semantics by using two features of defeasible reasoning including ambiguity blocking and propagating.

2.1 Defeasible Logic

Following the presentation in [7], a defeasible theory D is a triple $(F, R, >)$ where F is a finite set of facts, R is a finite set of rules, and $>$ is a superiority relation on R . The language of defeasible logic consists of a finite set of literals, l , and their complement $\sim l$.

A rule r in R is composed of an antecedent (body) $A(r)$ and a consequent (head) $C(r)$, where $A(r)$ consists of a finite set of literals and $C(r)$ contains a single literal. $A(r)$ can be omitted from the rule if it is empty. There are three types of rules in R , namely R_s (strict rules), R_d (defeasible rules), and R_{dft} (defeaters).

A conclusion derived from the theory D is a tagged literal and is categorised according to how the conclusion can be proved:

- $+\Delta q$: q is definitely provable in D .
- $-\Delta q$: q is definitely unprovable in D .
- $+\partial q$: q is defeasibly provable in D .
- $-\partial q$: q is defeasibly unprovable in D .

The provability is based on the concept of a derivation (or proof) in $D = (F, R, >)$. Informally, definite conclusions can be derived from strict rules by forward chaining, while defeasible conclusions can be obtained from defeasible rules iff all possible “attacks” are rebutted due to the superiority relation or defeater rules. A derivation is a finite sequence $P = (P(1), \dots, P(n))$ of tagged literals satisfying proof conditions (which correspond to inference rules for each of the four kinds of conclusions). $P(1..i)$ denotes the initial part of the sequence P of length i . In the follows, we present the proof conditions for definitely and defeasibly provable conclusions¹:

- $+\Delta$: If $P(i+1) = +\Delta q$ then
- (1) $q \in F$ or
 - (2) $\exists r \in R_s[q] \forall a \in A(r) : +\Delta a \in P(1..i)$
- $+\partial$: If $P(i+1) = +\partial q$ then either
- (1) $+\Delta q \in P(1..i)$ or
 - (2.1) $\exists r \in R_{sd}[q] \forall a \in A(r) : +\partial a \in P(1..i)$ and
 - (2.2) $-\Delta \sim q \in P(1..i)$ and
 - (2.3) $\forall s \in R_{sd}[\sim q]$ either
 - (2.3.1) $\exists a \in A(s) : -\partial a \in P(1..i)$ or
 - (2.3.2) $\exists t \in R_{sd}[q]$ such that $t > s$ and $\forall a \in A(t) : +\partial a \in P(1..i)$

The set of conclusions of a defeasible theory is finite², and it can be computed in linear time [10].

DL can be extended by an ambiguity propagating variant [11, 5]. The superiority relation is not considered in the inference process. Inference with the ambiguity propagation introduces a new tag Σ , a positive support for a literal $+\Sigma q$ is defined as:

$$+\Sigma: \text{ If } P(i+1) = +\Sigma q \text{ then } \exists r \in R_{sd}[q]: \forall a \in A(r) : +\Sigma a \in P(1..i)$$

$+\Sigma p$ means p is supported by the defeasible theory and there is a monotonic chain of reasoning that would lead us to conclude p in the absence of conflicts. A literal that is defeasibly provable ($+\partial$) is supported, but a literal may be supported even though it is not defeasibly provable. Thus support is a weaker notion than defeasible provability.

2.2 Argumentation Semantics

In what follows, we briefly introduce the basic notions of an argumentation system using defeasible reasoning. We also present the acceptance of an argument w.r.t. Dung’s semantics.

Definition 1. *An argument A for a literal p based on a set of rules R is a (possibly infinite) tree with nodes labelled by literals such that the root is labelled by p and for every node with label h :*

¹ For a full presentation and proof conditions of DL and its properties refer to [8, 9].

² It is the Herbrand base that can be built from the literals occurring in the rules and the facts of the theory.

1. If b_1, \dots, b_n label the children of h then there is a rule in R with body b_1, \dots, b_n and head h .
2. If this rule is a defeater then h is the root of the argument.
3. The arcs in a proof tree are labelled by the rules used to obtain them.

DL requires a more general notion of proof tree that admits infinite trees, so that the distinction is kept between an infinite un-refuted chain of reasoning and a refuted chain. Depending on the rules used, there are different types of arguments.

- A supportive argument is a finite argument in which no defeater is used.
- A strict argument is an argument in which only strict rules are used.
- An argument that is not strict, is called defeasible.

Relationships between two arguments A and B are determined by literals constituting these arguments. An argument A *attacks* a defeasible argument B if a literal of A is the complement of a literal of B , and that literal of B is not part of a strict sub-argument of B . A set of arguments \mathcal{S} *attacks* a defeasible argument B if there is an argument A in \mathcal{S} that attacks B .

A defeasible argument A is *undercut* by a set of arguments \mathcal{S} if \mathcal{S} supports an argument B attacking a proper non-strict sub-argument of A . An argument A is undercut by \mathcal{S} means some literals of A cannot be proven if we accept the arguments in \mathcal{S} .

The concepts of the attack and the undercut concern only defeasible arguments and sub-arguments. A defeasible argument is assessed as valid if we can show that the premises of all arguments attacking it cannot be proved from the valid arguments in \mathcal{S} . The concepts of provability depend on the method used by the reasoning mechanism to tackle ambiguous information. According to the features of the defeasible reasoning, we have the definition of acceptable arguments (definition 2).

Definition 2. An argument A for p is acceptable w.r.t. a set of arguments \mathcal{S} if A is finite, and

1. If reasoning with the ambiguity propagation is used: (a) A is strict, or (b) every argument attacking A is attacked by \mathcal{S} .
2. If reasoning with the ambiguity blocking is used: (a) A is strict, or (b) every argument attacking A is undercut by \mathcal{S} .

The status of an argument is determined by the concept of acceptance. If an argument can resist a reasonable refutation, this argument is justified. If an argument cannot overcome attacks from other arguments, this argument is rejected. We define the sets of justified arguments as follows:

Definition 3. Let D be a defeasible theory. We define J_i^D as follows.

- $J_0^D = \emptyset$
- $J_{i+1}^D = \{a \in \text{Args}^D \mid a \text{ is acceptable w.r.t. } J_i^D\}$

The set of justified arguments in a defeasible theory D is $J\text{Args}^D = \bigcup_{i=1}^{\infty} J_i^D$.

3 n-Person Argumentation Game

In this section, we utilise the argumentation semantics presented in section 2.2 to model agents' interactions in an n-person argumentation game. Also, we propose a knowledge structure which enables an agent to construct its arguments w.r.t. knowledge from other agents as well as to select a defensive argument.

3.1 Agents' Interactions

In an argumentation game, a group of agents \mathcal{A} shares a set of goals \mathcal{G} and a set of external constraints T_{bg} represented as a defeasible theory, known as a background knowledge. This knowledge provides common expectations and restrictions in \mathcal{A} . An agent has its own view on the working environment, therefore, can autonomously pursue its own goal. In this work, we model interactions between agents to settle on goals commonly accepted by the group. Also, at each step of the game, we show how an agent can identify a goal and sub-goals for its counter arguments. This information is critical for those agents whose main claims are refuted either directly by arguments from other agents or indirectly by the combination of these arguments.

Settling on Common Goals. An agent can pursue a goal in the set of common goals \mathcal{G} by proposing an explanation for its goal. The group justifies proposals from individual agents in order to identify commonly-accepted goals using a dialogue as follows:

1. Each agent broadcasts an argument for its goal. The system can be viewed as an argumentation game with n-players corresponding to the number of agents.
2. An agent checks the status of its argument against those from the other agents. There are three possibilities: (a) *directly refuted* if its argument conflicts with those from others; (b) *collectively refuted* if its argument does not conflict with individual arguments but violates the combination of individual arguments (See section 3.2); (c) *collectively accepted* if its argument is justified by the combination (See section 3.3).
3. According to the status of its main claim, an agent can: (a) defend its claim; (b) attack a claim from other agents; (c) rest. An agent repeats the previous step until the termination conditions of the game are reached.
4. The dialogue among agents is terminated if all agents can pass their claims. For a dispute, agents stop arguing if they do not have any more argument to propose.

Weighting Opposite Premises. In a dialogue, at each iteration, an agent is required to identify goals and sub-goals which are largely shared by other agents. This information is highly critical for agents, whose main claims are refuted either directly by other agents or collectively by the combination of arguments from others in order to effectively convince other agents.

To achieve that an agent, A_{me} , identifies a sub-group of agents, namely "opp-group", which directly or collectively attacks its main claim. A_{me} creates $Args^{opp}$ as the set of opposing arguments from the opp-group and P^{opp} as the set of premises in $Args^{opp}$. Essentially, $Args^{opp}$ contains arguments attacking A_{me} 's claim. Each element of P^{opp} is weighted by its frequency in $Args^{opp}$. We define the preference over P^{opp} as given

$p_1, p_2 \in P^{\text{opp}}$, $p_2 \succeq p_1$ if the frequency of p_2 in $Args^{\text{opp}}$ is greater than that of p_1 . Basically, the more frequent an element $q \in P^{\text{opp}}$ is the more agents use this premise in their arguments. Therefore the refutation of q challenges other agents better than the premises having lower frequency since this refutation causes a larger number of agents to reconsider their claims.

Defending the Main Claim. At iteration i , $Args_i^{\text{opp}}$ represents the set of arguments played by the opp-group:

$$Args_i^{\text{opp}} = \bigcup_{j=0}^{|\mathcal{A}|} Args_i^{A_j} | Args_i^{A_j} \text{ directly attacks } Args_i^{A_{me}}$$

where $Args^{A_j}$ is the argument played by agent A_j . If A_j rests at iteration i , its last argument (at iteration k) is used $Args_i^{A_j} = Args_k^{A_j}$. The set of opposite premises at iteration i is:

$$P_i^{\text{opp}} = \{p | p \in Args_i^{\text{opp}} \text{ and } p \notin Args_i^{A_{me}}\}$$

The preference over elements of P^{opp} provides a mechanism for A_{me} to select arguments for defending its main claim.

Example 1. Suppose that agent A_1 and A_2 respectively propose $Args^{A_1} = \{\Rightarrow e \Rightarrow b \Rightarrow a\}$ and $Args^{A_2} = \{\Rightarrow e \Rightarrow c \Rightarrow a\}$ whilst agent A_3 claims $Args^{A_3} = \{\Rightarrow d \Rightarrow \sim a\}$. From A_3 's view, its claim directly conflicts with those of A_1 and A_2 . The arguments and premises of the opp-group are:

$$Args_i^{\text{opp}} = \{\Rightarrow e \Rightarrow b \Rightarrow a; \Rightarrow e \Rightarrow c \Rightarrow a\} \text{ and } P_i^{\text{opp}} = \{a^2, b^1, c^1, e^2\}$$

The superscript of elements in P_i^{opp} represents the frequency of a premise in $Args_i^{\text{opp}}$. A_3 can defend its claim by providing a counter-argument that refute $\sim a$ – the major claim of the opp-group. Alternatively, A_3 can attack either b or c or e in the next step. An argument against e is the better selection compared with those against b or c since A_3 's refutation of e causes both A_1 and A_2 to reconsider their claims.

Attacking an Argument. In this situation, individual arguments of other agents do not conflict with that of A_{me} but the integration of these arguments does. Agent A_{me} should argue against one of these arguments in order to convince others about its claim.

At iteration i , let the integration of arguments be $T_{\text{INT}}^i = T_{\text{bg}} \cup_{j=0}^{|\mathcal{A}|} T_j^i$, where T_j^i is the knowledge from agent j supporting agent j 's claim, and $JArgs_{\text{INT}}^i$ be the set of justified arguments from integrated knowledge of other agents (See section 3.3). The set of opposite arguments is defined as:

$$Args_i^{\text{opp}} = a | a \in JArgs_{\text{INT}}^i \text{ and } a \text{ is attacked by } Args_i^{A_{me}}$$

and the set of opposite premises is:

$$P_i^{\text{opp}} = \{p | p \in Args_i^{\text{opp}} \text{ and } (p \notin Args_i^{A_{me}} \text{ or } p \text{ is not attacked by } Args_i^{A_{me}})\}$$

The second condition is to guarantee that A_{me} is self-consistent and does not play any argument against itself. In order to convince other agents about its claim, A_{me} is required to provide arguments against any premise in P^{opp} . In fact, the order of elements in P^{opp} offers a guideline for A_{me} on selecting its attacking arguments.

3.2 Agent's Knowledge Structure

In this section, we present a knowledge structure which allows an agent to incorporate background knowledge and knowledge exposed by individual agents during the game. Also, we propose two simple methods to integrate knowledge sources w.r.t. ambiguity information.

Knowledge Representation. Agent A_{me} has three types of knowledge including the background knowledge T_{bg} , its own knowledge about working environment T_{me} , and the knowledge about others:

$$\mathcal{T}_{other} = \{T_j : 1 \leq j \leq |\mathcal{A}| \text{ and } j \neq me\}$$

where T_j is obtained from agent A_j during iterations and T_j is represented in DL. At iteration i , the knowledge obtained from A_j is accumulated from previous steps:

$$T_j^i = \bigcup_{k=0}^{i-1} T_j^k + Args_i^{A_j}$$

In our framework, the knowledge of an agent can be rebutted by other agents. It is reasonable to assume that defeasible theories contain only defeasible rules and defeasible facts (defeasible rules with empty body).

Knowledge Integration. To generate arguments, an agent integrates knowledge from different sources. Given ambiguous information between two sources, there are two possible methods to combine them: ambiguity blocking is selected if the preference between these sources is known; otherwise, ambiguity propagation is applied.

Ambiguity Blocking Integration. This method extends the standard defeasible reasoning by creating a new superiority relation from that of the knowledge sources i.e. given two sources as T_{sp} – the superior theory, and T_{in} – the inferior theory, we generate a new superiority relation $R_d^{sp} > R_d^{in}$ based on rules from two sources. The integration of the two sources is denoted as $T_{INT} = T_{sp} \ni T_{in}$. Now, the standard defeasible reasoning can be applied for T_{INT} to produce a set of arguments $Args_{AB}^{T_{INT}}$.

Example 2. Given two defeasible theories

$$\begin{aligned} T_{bg} &= \{R_d = \{r_1 : e \Rightarrow c; r_2 : g, f \Rightarrow \sim c, r_3 : \Rightarrow e\}; > = \{r_2 > r_1\}\} \\ T_{me} &= \{R_d = \{r_1 : \Rightarrow d; r_2 : d \Rightarrow \sim a; r_3 : \Rightarrow g\}\} \end{aligned}$$

The integration of $T_{bg} \ni T_{me}$ produces:

$$\begin{aligned} T_{INT} &= \{R_d = \{r_1^{T_{bg}} : e \Rightarrow c; r_2^{T_{bg}} : g, f \Rightarrow \sim c, r_3^{T_{bg}} : \Rightarrow e; r_1^{T_{me}} : \Rightarrow d; r_2^{T_{me}} : d \Rightarrow a; r_3^{T_{me}} : \Rightarrow g\}; \\ &> = \{r_2^{T_{bg}} > r_1^{T_{bg}}\}\} \end{aligned}$$

The integrated theory inherits the superiority relation from T_{bg} . That means the new theory reuses the blocking mechanism from T_{bg} .

Ambiguity Propagation Integration. Given two knowledge sources T_1 and T_2 , the reasoning mechanism with ambiguity propagation can directly apply to the combination of theories denoted as $T'_{\text{INT}} = T_1 + T_2$. The preference between two sources is unknown, therefore, there is no method to solve conflicts between them. The supportive and opposing arguments for any premise are removed from the final set of arguments. The set of arguments obtained by this integration is denoted by $Args_{\text{AP}}^{T'_{\text{INT}}}$.

3.3 Argument Justification

The motivation of an agent to participate in the game is to promote its own goal. However, its claim can be refuted by different agents. To gain the acceptance of the group, at the first iteration, an agent should justify its arguments by common constraints and expectations of the group governed by the background knowledge T_{bg} . The set of arguments justified by T_{bg} determines arguments that an agent can play to defend its claim. In subsequent iterations, even if the proposal does not conflict with other agents, an agent should ponder knowledge from others to determine the validity of its claim. That is an agent is required a justification by collecting individual arguments from others.

Justification by Background Knowledge. Agent A_{me} generates the set of arguments for its goals by combining its private knowledge T_{me} and the background knowledge T_{bg} . The combination is denoted as $T_{\text{INT}} = T_{\text{bg}} \oplus T_{\text{me}}$ and the set of arguments is $Args^{T_{\text{INT}}}$. Due to the non-monotonic nature of DL, the combination can produce arguments beyond individual knowledges. From A_{me} 's view, this can bring more opportunities to fulfil its goals. However, A_{me} 's arguments must be justified by the background knowledge T_{bg} since T_{bg} governs essential behaviours (expectations) of the group. Any attack to T_{bg} is not supported by members of \mathcal{A} . A_{me} maintains the consistency with the background knowledge T_{bg} by following procedure:

1. Create $T_{\text{INT}} = T_{\text{bg}} \oplus T_{\text{me}}$. The new defeasible theory is obtained by replicating all rules from common constraints T_{bg} into the internal knowledge T_{me} while maintaining the superiority of rules in T_{bg} over that in T_{me} .
2. Use the ambiguity blocking feature to construct the set of arguments $Args^{T_{\text{bg}}}$ from T_{bg} and the set of arguments $Args_{\text{AB}}^{T_{\text{INT}}}$ from T_{INT} .
3. Remove any argument in $Args_{\text{AB}}^{T_{\text{INT}}}$ attacked by those in $Args^{T_{\text{bg}}}$, obtaining the justified arguments by the background knowledge $JArgs^{T_{\text{INT}}} = \{a \in Args_{\text{AB}}^{T_{\text{INT}}} \text{ and } a \text{ is accepted by } Args^{T_{\text{bg}}}\}$.

Example 3. Consider two defeasible theories:

$$T_{\text{bg}} = \{R_{\text{d}} = \{r_1 : e \Rightarrow c; r_2 : g, f \Rightarrow \sim c, r_3 : \Rightarrow e\}; > = \{r_2 > r_1\}\}$$

$$T_{\text{me}} = \{R_{\text{d}} = \{r_1 : \Rightarrow d; r_2 : d \Rightarrow \sim a; r_3 : \Rightarrow g\}\}$$

We have sets of arguments from the background theory and the integrated theory:

$$Args^{T_{\text{bg}}} = \{\Rightarrow e; \Rightarrow e \Rightarrow c\}$$

$$Args^{T_{\text{INT}}} = Args^{T_{\text{bg}} \oplus T_{\text{me}}} = \{\Rightarrow e; \Rightarrow e \Rightarrow c; \Rightarrow d; \Rightarrow g; \Rightarrow d \Rightarrow \sim a\}$$

In this example, there is not any attack between arguments in $Args^{T_{bg}}$ and $Args_{AB}^{T_{INT}}$. In other words, arguments from $Args^{T_{INT}}$ are acceptable by those from $Args^{T_{bg}}$. The set of justified arguments w.r.t. $Args^{T_{bg}}$ is $JArgs^{T_{INT}} = Args_{AB}^{T_{INT}}$.

Collective Justification. During the game, A_{me} can exploit the knowledge exposed by other agents in order to defend its main claims. Due to possible conflicts in individual proposals, an agent uses the sceptical semantics of the ambiguity propagation reasoning to retrieve the consistent knowledge. Essentially, given competing arguments an agent does not have any preference over them, therefore, these arguments will be rejected. The consistent knowledge from the others allows an agent to discover ‘‘collective wisdom’’ distributed among agents in order to justify its claim.

The justification of collective arguments, which are generated by integrating all knowledge sources, is done by the arguments from the background knowledge $Args^{T_{bg}}$. The procedure runs as follows:

1. Create a new defeasible theory $T'_{INT} = T_{bg} \ni T_{me} + \mathcal{T}_{other}$.
2. Generate the set of arguments $Args'_{AP}{}^{T'_{INT}}$ from T'_{INT} using the feature of ambiguity propagation.
3. Justify the new set of arguments $JArgs'_{INT} = \{a | a \in Args'_{AP}{}^{T'_{INT}} \text{ and } a \text{ is accepted by } Args^{T_{bg}}\}$.

$JArgs'_{INT}$ allows A_{me} to verify the status of its arguments for its claim $JArgs^{T_{INT}}$. If arguments in $JArgs'_{INT}$ and $JArgs^{T_{INT}}$ do not attack one another, A_{me} 's claims are accepted by other agents. Any conflict between two sets shows that accepting arguments in $JArgs'_{INT}$ stops A_{me} to achieve its claims in next steps. The set of arguments $Args^{OPP}$ against A_{me} is identified as any argument in $JArgs'_{INT}$ attacking A_{me} 's arguments. A_{me} also establishes P^{OPP} to select its counter-argument. It is noticed that A_{me} is self-consistent.

Example 4. Suppose the background knowledge T_{bg} and the private knowledge T_{me} of A_{me} are:

$$\begin{aligned} T_{bg} &= \{R_d = \{r_1 : e \Rightarrow c; r_2 : g, f \Rightarrow \sim c\}; > = \{r_2 > r_1\}\} \\ T_{me} &= \{R_d = \{r_1 : \Rightarrow e; r_2 : c \Rightarrow d; r_3 : \Rightarrow g\}\} \end{aligned}$$

Agent A_{me} currently plays $\{\Rightarrow e \Rightarrow c \Rightarrow d\}$ and knows about other agents:

$$\mathcal{T}_{other} = \{T_1, T_2\} \text{ where } T_1 = \{\Rightarrow h \Rightarrow f \Rightarrow b \Rightarrow a\} \text{ and } T_2 = \{\Rightarrow e \Rightarrow c \Rightarrow a\}$$

The claim of A_3 is acceptable w.r.t. arguments played by the other agents. However, the combination $T'_{INT} = T_{bg} \ni T_{me} + \mathcal{T}_{other}$ shows the difference. This combination generates $\{\Rightarrow g, \Rightarrow e, \Rightarrow e \Rightarrow f \Rightarrow b, \Rightarrow g, f \Rightarrow \sim c\}$. $\{\Rightarrow g, f \Rightarrow \sim c\}$ is due to the superiority relation in T_{bg} which rebuts the claim of A_3 . Therefore, the set of opposing arguments $Args^{OPP} = \{\Rightarrow g, f \Rightarrow \sim c\}$ and $P^{OPP} = \{f^1\}$. Given this information, A_3 should provide a counter-evidence to f in order to pursue c . Moreover, A_3 should not expose g to the other agents. Otherwise, A_3 has to drop its initial claim d .

4 Related Works

Substantial works have been done on argumentation games in the artificial intelligence and law-field. [1] introduces a dialectical model of legal argument, in the sense that arguments can be attacked with appropriate counterarguments. In the model, the factual premises are not arguable; they are treated as strict rules. [12] presents an early specification and implementation of an argumentation game based on the Toulmin argument-schema without a specified underlying logic. [13] presented the pleadings game as a normative formalization and fully implemented computational model, using conditional entailment.

Settling on a common goal among agents can be seen as a negotiation process where agents exchange information to resolve conflicts or to obtain missing information. The work in [14] provides a unified and general formal framework for the argumentation-based negotiation dialogue between two agents. The work establishes a formal connection between the status of an argument (accepted, rejected, and undecided) with an agent's actions (accept, reject, and negotiate respectively). Moreover, an agent's knowledge is evolved by accumulating arguments during interactions.

[3] presents an argumentation-based coordination, where agents can exchange arguments for their goals and plans to achieve the goals. The acceptance of an argument of an agent depends on the attitudes of this agent namely credulous, cautious, and sceptical. In [15], agents collaborate with one another by exchanging their proposals and counter-proposals in order to reach a mutual agreement. During conversations, an agent can retrieve missing literals (regarded as sub-goals) or fulfil its goals by requesting collaboration from other agents.

We have advantages of using DL since it flawlessly captures the statuses of arguments, such as accepted, rejected, and undecided by the proof conditions of DL. The statuses are derived from the notions of $+\partial$, $-\partial$ and $+\Sigma$ corresponding to a positive proof, a negative proof, and a positive support of a premise. Consequently, an agent can take a suitable action either to provide more evidence or to accept an argument from others. In addition, DL provides a compact representation to accommodate new information.

Using DL to capture concepts of the argumentation game is supported by [16, 17] and recently [18]. [16] focuses on persuasive dialogues for cooperative interactions among agents. It includes in the process cognitive states of agents such as knowledge and beliefs, and presents some protocols for some types of dialogues (e.g. information seeking, explanation, persuasion). [17] provides an extension of DL to include the step of the adversarial dialogue by defining a meta-program for an alternative computational algorithm for ambiguity propagating DL while the logic presented here is ambiguity blocking. In [18], arguments are generated by using the defeasible reasoning with ambiguity blocking. After each step in an argumentation game, an agent can upgrade the strength of its arguments if these arguments are not refuted by the opposing agent.

We tackle the problem of evolving knowledge of an agent during iterations, where the argument construction is an extension of [18]. In our work, we define the notion of collective acceptance for an argument and a method to weight arguments defending against opposing arguments by using both features of ambiguity blocking and propagating.

The works in literature did not clearly show how an agent can tackle with conflicts from multiple agents, especially when the preference over arguments is unknown. The main difference in our framework is the external model where more than two agents can argue to settle on goals commonly accepted by the group. Our weighting mechanism enables an agent to build up a preference over premises constituting opposing arguments from other agents. As a result, an agent can effectively select an argument among those justified by the group's background knowledge to challenge other agents.

We also propose the notion of collective justification to tackle the side-effect of accepting claims from individual agents. Individual arguments for these claims may not conflict with one another, but the integration of these arguments can result in conflicting with an agent's claim. This notion is efficiently deployed in our work due to the efficiency of defeasible logic in handling ambiguous information.

5 Conclusions

We presented an n-person argumentation game based on defeasible logic, which enables a group of more than two agents to settle on goals commonly accepted by the group. During an argumentation game, each agent can use knowledge from multiple sources including the group's constraints and expectations, other agents' knowledge, and its own knowledge in order to argue to convince other agents about its goals. The knowledge about the group's constraints and expectations plays a critical role in our framework since this knowledge provides a basis to justify new arguments non-monotonically inferred from the integration of different sources.

In this work, we propose a simple weighting mechanism, which is based on the frequency of premises in arguments attacking an agent's claim, in order to tackle the problem of conflicts from multiple agents. In the future work, we will extend this mechanism to incorporate the notion of trustful arguments from trusted agents to better select a rebuttal argument and resolve conflicts among agents.

References

- [1] Prakken, H., Sartor, G.: A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law* 4, 331–368 (1996)
- [2] Jennings, N.R., Parsons, S., Noriega, P., Sierra, C.: On argumentation-based negotiation. In: *Proceedings of the International Workshop on Multi-Agent Systems*, pp. 1–7 (1998)
- [3] Parsons, S., McBurney, P.: Argumentation-based dialogues for agent coordination. *Group Decision and Negotiation* (12), 415–439 (2003)
- [4] 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), 321–358 (1995)
- [5] Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: A flexible framework for defeasible logics. In: *Proc. American National Conference on Artificial Intelligence*, pp. 401–405 (2000)
- [6] Maher, M.J., Rock, A., Antoniou, G., Billington, D., Miller, T.: Efficient defeasible reasoning systems. *International Journal of Artificial Intelligence Tools* 10(4), 483–501 (2001)

- [7] Billington, D.: Defeasible logic is stable. *Journal of Logic and Computation* 3, 370–400 (1993)
- [8] Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. *ACM Transactions on Computational Logic* 2(2), 255–287 (2001)
- [9] Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Embedding defeasible logic into logic programming. *Theory and Practice of Logic Programming* 6(6), 703–735 (2006)
- [10] Maher, M.J.: Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming* 1(6), 691–711 (2001)
- [11] Governatori, G., Maher, M.J., Antoniou, G., Billington, D.: Argumentation Semantics for Defeasible Logic. *J. Logic Computation* 14(5), 675–702 (2004)
- [12] Bench-Capon, T.J.: Specification and implementation of Toulmin dialogue game. In: Hage, J.C., Bench-Capon, T.J.M., Koers, A.W., de Vey Mestdagh, C.N.J., Grutters, C.A.F.M. (eds.) *Jurix* 1998, pp. 5–20 (1998)
- [13] Lodder, A.R.: Thomas F. Gordon, The Pleadings Game – an artificial intelligence model of procedural justice. *Artif. Intell. Law* 8(2/3), 255–264 (2000)
- [14] Amgoud, L., Dimopoulos, Y., Moraitis, P.: A unified and general framework for argumentation-based negotiation. In: *Proceedings of the 6th international joint conference on AAMAS*, pp. 1–8 (2007)
- [15] Rueda, S.V., Garcia, A.J., Simari, G.R.: Argument-based negotiation among bdi agents. *Journal of Computer Science and Technology* 2(7), 1–8 (2002)
- [16] Letia, I.A., Vartic, R.: Defeasible protocols in persuasion dialogues. In: *WI-IATW 2006: Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*, pp. 359–362 (2006)
- [17] Hamfelt, A., Eriksson, J., Nilsson, J.F.: A metalogic formalization of legal argumentation as game trees with defeasible reasoning. In: *ICAIL 2005: Proceedings of the 10th international conference on Artificial intelligence and law*, pp. 250–251. ACM, New York (2005)
- [18] Thakur, S., Governatori, G., Padmanabhan, V., Eriksson Lundström, J.: Dialogue games in defeasible logic. In: Orgun, M.A., Thornton, J. (eds.) *AI 2007. LNCS (LNAI)*, vol. 4830, pp. 497–506. Springer, Heidelberg (2007)

Revenue Maximising Adaptive Auctioneer Agent

Janine Claire Pike and Elizabeth Marie Ehlers

Academy for Information Technology, APK Campus,
University of Johannesburg, Johannesburg, South Africa
janine.pike@gmail.com

Abstract. Auction theory has proven that auction revenue is influenced by factors such as the auction format and the auction parameters. The Revenue Maximising Adaptive Auctioneer (RMAA) agent model has been developed with the aim of generating maximum auction revenue by adapting the auction format and parameters to suit the auction environment. The RMAA agent uses a learning classifier system to learn which rules are profitable in a particular bidding environment. The profitable rules are then exploited by the RMAA agent to generate maximum revenue. The RMAA agent model can effectively adapt to a real time dynamic auction environment.

Keywords: Agent auctions, auction theory, reinforcement learning, learning classifier system, ZCS.

1 Introduction

An auction is a rule-based mechanism that is used to allocate resource(s) amongst a set of competing bidders. Auctioneer or seller agents will want to maximize their revenue in an auction. Auction parameters such as the auction format, the reserve price, the closing of the auction etc. will directly influence the number of bidders as well as the nature of bidders that will participate in the auction [1] and thus the revenue of the seller. The importance of auction design can be illustrated by the New Zealand radio spectrum auction in 1990. The projected revenue for the auction was NZ\$250 million but the auction only managed to generate a mere NZ\$36 million. The auction format selected was a Vickrey auction where a Dutch auction would have generated considerably more revenue [2]. The aim of this research is to develop an auctioneer agent that maximises auction revenue by selecting the most suitable auction mechanism and auction parameters based on the environment and bidder characteristics.

There are four standard single-unit auction types: the English auction, the Dutch auction, the first price sealed bid auction and the Vickrey auction. The Dutch and first price sealed bid auction are known as first price auctions. The English and Vickrey auctions are known as second price auctions. Several factors influence the profitability of an auction format:

- The risk profile of a bidder influences his bidding behaviour. In first price auctions, a risk averse bidder will raise his bid slightly to gain a higher probability of

winning the auction at the cost of a reduced profit. If the bidders are risk averse first price auctions will be more profitable [3]. Conversely, if bidders are risk seeking then second price auctions will generate more revenue [4].

- In auctions where the item is a common value or correlated value item and there are at least three bidders the English auction will produce a greater profit for the auctioneer than the Vickrey auction, which in turn will produce a greater profit than the first price auctions. The English auction is the most profitable as the common value component of a bidder's valuation can be influenced by the valuations of the other bidders [5].
- If bidders are symmetric then they draw their valuation from the same distribution. If bidders are asymmetric they draw their valuations from different distributions. It is difficult to rank the profitability of auctions based on asymmetry as different variations of asymmetry will each have a different effect on auction revenue [3].

Several experiments have been conducted using online auctions to investigate the factors that influence the profitability of internet auctions. The experiments yielded the following results [6, 7, 8]:

- The reserve price increases auction revenue but decreases the probability of a sale. A reserve price is the minimum price the seller will accept for an item.
- The Dutch auction generated 30% more revenue than the first price sealed bid auction.
- A high initial price and a longer auction duration results in greater auction revenue.
- A soft auction close is more profitable than a hard auction close. A hard close has a fixed deadline whereas the deadline is extended in a soft close if a bid is placed in the closing minutes of the auction.

2 Related Work

Gregg and Walczak have developed an Auction Advisor agent that uses historical data to make recommendations to both buyers and sellers. The Auction Advisor collects related data from auction sites and derives recommendations from analysing the collected data and auction research [9]. Pardoe and Stone have developed an autonomous adaptive auction mechanism that adjusts the auction parameters in response to past results. The auction parameters include the reserve price, the auction close, the auctioneer fees and the minimum bid increment. The adaptive mechanism uses reinforcement learning to determine the optimal auction parameters. In a simulation comparing the adaptive auction mechanism to a fixed auction mechanism the adaptive auction mechanism generated greater revenue than any fixed parameter choice [10]. Cliff uses an evolutionary approach where the auction mechanism is defined by the parameter Q_s . At each interval, Q_s is the probability that a seller will be chosen to quote. If $Q_s = 0.5$ the auction mechanism is equivalent to a continuous double auction. A genetic algorithm is used to explore the space of auction mechanisms. The fitness function used by Cliff measured the efficiency of the auction mechanism this however could easily be adjusted to measure auction revenue. Cliff's experimental results found that new hybrid auction mechanisms can outperform traditional auction mechanisms in some environments [11].

3 RMAA Model

The Revenue Maximising Adaptive Auctioneer (RMAA) agent model is an adaptive, autonomous agent that selects the auction format and parameters based on an evaluation of the environment and previous auction results. Heuristics from auction theory and results from internet experiments are used to guide the selection of auction parameters. The RMAA model consists of an Auction Manager and an Auction Initialisation Module. The Auction Initialisation Module is composed of an Auction Format Selector (AFS) Component and an Auction Parameter Selector (APS) Component.

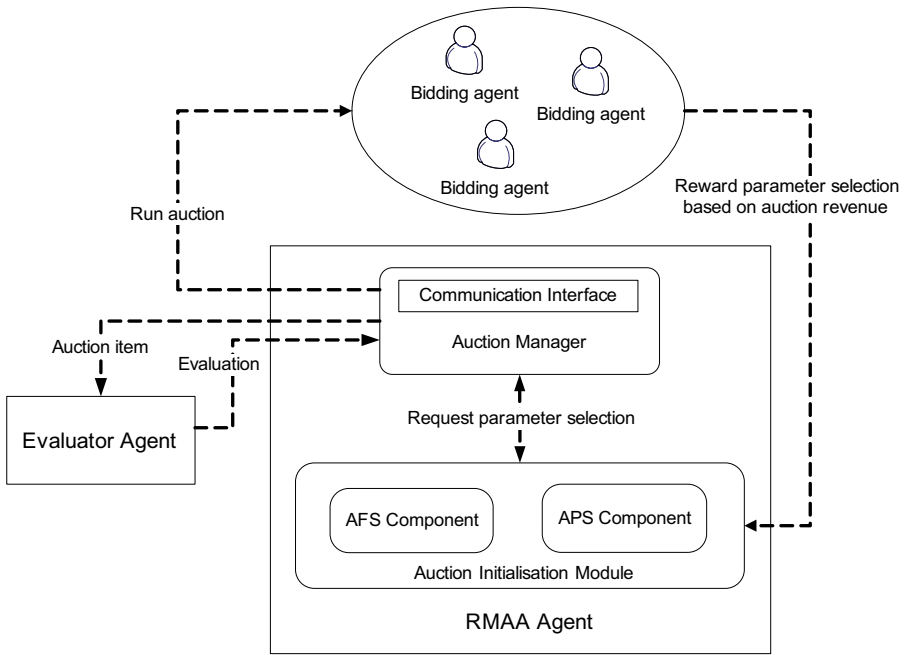


Fig. 1. The RMAA agent model

3.1 The Auction Manager

The Auction Manager automates the running of the auction. The Auction Manager will create a new auction using the parameters determined by the Auction Initialisation Module. The Auction Manager will communicate with the bidders, validate any incoming bids, record the bid history of the auction and determine the winning bidder.

3.2 Auction Initialisation Module

The Auction Initialisation Module is responsible for selecting the parameters that will define the auction mechanism. The AFS Component will determine the following as necessary: the auction format (English, Dutch, Vickrey or First Price Sealed Bid), if a

reserve price will be used, if a buy out price will be used and the type of auction close. The APS Component will determine the value of following parameters as necessary: the reserve price, the auction duration, the buyout price, the initial starting price and the bid increment.

3.2.1 AFS Component

The AFS Component is implemented using a learning classifier system, more specifically the Zeroth Level Classifier System (ZCS). A ZCS provides a way to exploit the existing knowledge about maximising auction revenue while simultaneously discovering new knowledge. A ZCS consists of a database of classifiers which are rules of the form *if condition then action*. Rules whose conditions match the current condition of the environment are placed in the current match set [M]. The classifiers are then grouped according to action and roulette wheel selection is used to select the action. The stronger the fitness of the group, the higher the probability the action of the group will be chosen. All classifiers with the chosen action are placed in the action set [A]. The strength of a classifier is the predictor of the reward it should receive. If the reward received is greater than or less than the classifier strength, the classifier strength should be updated accordingly. The ZCS introduces new classifiers through the use of a genetic algorithm as a means of exploring the solution space. The execution of the genetic algorithm is controlled by the parameter P . On any given turn there is a probability of P that the genetic algorithm will be invoked. When the genetic algorithm is invoked, two classifiers will be selected from the classifier population. Two new classifiers will be created by copying, possibly crossing and mutating the two selected classifiers. The classifier population must always be kept constant therefore the two weakest classifiers must be deleted [12].

The AFS has a database of classifiers. The condition portion of a classifier indicates information regarding the bidder environment. The action portion of a classifier indicates the auction format to be used. The database of classifiers will contain heuristics from auction theory and internet experiments. The AFS requires a description of the bidder environment from the Evaluator agent. The Evaluator agent will determine information such as the risk profile of bidders and whether the auction is a common value or private value auction. The Evaluator agent will implement the model proposed by Zhang et al. to determine the probable risk profile of the typical bidder that the auction will attract based on the characteristics of the item for sale. Zhang et al. derived the following formula to determine the risk profile of potential bidders [13]:

$$Y = 1/6 * HI/50000 + 1/6 * PR/RA - 1/6 * DU/5 + \epsilon . \quad (1)$$

In formula (1) the potential mean household income required to purchase the item (HI), the price of the item (PR), the highest price for an item in the same category (RA) and the durability of the item (DU) are used to determine risk profile. If y is in the range (0.495, 0.505) the bidders will be risk neutral, if y is greater than 0.505 the bidders will be risk seeking and if y is less than 0.495 bidders will be risk averse [13].

The Evaluator agent can also use the item for sale to determine whether the auction will be a common or a private value auction. In private value auctions each bidder's valuation for the item is based on their own personal preferences. In common value

auctions there is only one value for the item and each bidder will have their own estimate of this value based on their own information. For example a painting purchased for personal enjoyment only will be a private value auction and a piece of land with unknown oil reserves will be a common value auction [3]. The Evaluator agent will send a description of the probable auction environment to the Auction Manager. The Auction Manager will provide the AFS Component with this description when it requests the auction format and parameters to be used. The AFS Component will compare the description against the conditions of the classifiers in the rule base. The ZCS will then determine which auction format to use and will request any other required parameters from the APS.

3.2.2 The APS Component

The APS Component must select the optimal value for parameters such as the reserve price. The reserve price will have a range of possible values. For example it could be set to 50, 70, 100 or 110 percent of the item's book value. The APS Component needs to determine over time which is the most profitable value. The problem of selecting the most profitable value over time is a k-armed bandit problem. The k-armed bandit problem revolves around finding the right balance between exploitation and exploration. There are several algorithms that can be used to solve the k-armed bandit problem. The ϵ -greedy method will be chosen as it can significantly outperform several of the more complicated methods [14]. Using the ϵ -greedy method a random parameter value will be chosen ϵ percent of the time. The parameter value with the highest action value (the reward associated with a particular parameter value) will be chosen $1-\epsilon$ percent of the time.

The RMAA agent model provides several benefits over related research. Firstly it automates the auction on behalf of the seller instead of just providing recommendations to the seller. The RMAA agent is designed for a dynamic, real-time auction environment instead of only for a simulation environment. The RMAA model uses universally known auction formats and thus embedding strategies into bidding agents is simple as optimal bidding strategies are well known. The classifier database contains heuristics from auction theory and internet experiments providing the agent with valuable knowledge. The heuristics enable the auctioneer agent to be effective immediately instead of having to wait for the agent to first learn useful classifiers before it becomes effective. The RMAA agent is not based on assumptions regarding the bidder population instead it aims to adapt to any bidder population. The RMAA model implements evolutionary learning that will create new classifiers that were not originally in the classifier database.

4 Implementation

A simulation based on a prototype of the RMAA agent model has been implemented using the Java Agent DEvelopment Framework (JADE). A population size of 20 classifiers for the AFS has been chosen. The structure of the classifier is defined according to the table below.

Table 1. A description of the AFS classifier

Auction value model	Bidder risk profile	Auction format	Reserve Price	Auction close
Private: 0 Common: 1	Neutral: 00 Averse: 01 Seeking: 10	English: 00 Dutch: 01 FPSB: 10 Vickrey: 11	No: 0 Yes: 1	Soft: 0 Hard: 1

The auction value model and the bidder risk profile describe the condition portion of the classifier. The auction format, reserve price and auction close describe the action portion of the classifier. Several classifiers based on heuristics from auction theory and internet experiments have been inserted into the classifier rule base. The remaining required classifiers have been randomly generated. The classifiers based on heuristics are assigned a higher initial strength than the randomly generated classifiers as they should generate greater revenue. Bidding agents are required as part of the simulation. Each bidding agent is assigned a valuation drawn from a uniform probability distribution. The bid value for each agent will be determined using one of the following Nash equilibrium formulas [13]:

$$b_i = (n-1) / (n) * v_i . \quad (2)$$

$$b_i = (n-1) / (n-1 + r_i) * v_i . \quad (3)$$

Formula (2) is the bid function for a risk neutral bidder. Formula (3) is the bid function for risk averse and risk seeking bidders. n is the number of bidders, v_i is the valuation of bidder i , r_i is the risk profile of bidder i . A value between 0 and 1 for r_i indicates a risk averse bidder and a value greater than 1 indicates a risk seeking bidder.

JADE has a Directory Facilitator that functions as a yellow pages service enabling the bidding agents to locate and register with the auctioneer agent. Registering with the auctioneer agent simply entails sending the auctioneer agent their unique identifier so that the bidding agents can be notified of any auctions. JADE facilitates message passing between agents using messages conforming to the FIPA ACL specification. ACL messages such as inform, propose, accept proposal, reject proposal are naturally suited to the communication required in an auction. A GUI captures the settings required for the auction simulation such as the predominant bidder risk profile, the number of bidders and whether the auction is common value or private value. The Auction Manager creates an auction for an item according to the specification provided by the Auction Initialisation Module. Once the auction has been created an ACL inform message is sent to all potential bidders. The bidders will participate in the auction if their valuation is greater than the reserve price. Once the auction is complete the classifiers in [A] are rewarded using the following formula derived from [12]:

$$C_s = C_s + \beta((\text{bid}_{\text{win}}/\text{bk}_{\text{value}}) - C_s) . \quad (4)$$

Formula (4) updates the classifier strength using β to control the degree of impact a new reward has on the classifier strength. β is assigned an initial value of 0.5 to ensure that if any initial classifier strengths were inaccurate they will be quickly corrected. B then steadily decreases to 0.2 over several auction simulations. bid_{win} is the winning bid and bk_{value} is the book value of the item.

The parameters controlled by the APS Component are the starting price, the reserve price, the minimum bid increment and the auction duration. The value of ϵ in the APS Component is assigned a high initial value of 0.5 which will slowly decrease to a constant value of 0.1 over several auction runs. The high initial value of ϵ will encourage significant exploration in the early stages which will taper off as more auction simulations are run. An element of exploration will always remain. The reason for this is that as the bidder population varies over time the action values for each parameter will change. The APS Component exploits heuristics from internet experiments by using the information to guide the initialisation of the action values for each parameter. For example a high starting price leads to greater revenue so the action values of the higher starting prices will be initialized to a greater value than the action values of lower starting prices.

5 Conclusion and Future Research

Auction theory as well as online auction experiments have proven several heuristics that are useful in designing an auction mechanism that will result in maximum profitability. No auction mechanism can perform optimally in all environments an auctioneer agent must thus be able to adapt to different auction environments. The RMAA agent uses reinforcement learning and evolutionary learning in order to achieve adaptability. The RMAA agent will select the optimal auction parameters based on the environment. The RMAA model is based on the standard single unit auctions namely the English, Dutch, Vickrey and first price sealed bid auctions. Similarly to single unit auctions, the revenue of multi-unit auctions is also affected by certain factors. The RMAA model could be extended to include multi-unit auctions as well. Further research would also be required to develop and implement the Evaluator agent that must provide the RMAA agent with the required information about the bidding environment based on the auction item for sale. The ability of the RMAA agent model to exploit proven heuristics regarding auction profitability provides several benefits over related research.

References

1. Gerding, E.H., Rogers, A., Dash, R.K., Jennings, N.R.: Competing Sellers in Online Markets: Reserve Prices, Shill bidding, and Auction Fees. In: 5th International Joint Conference on Autonomous agents and Multiagent Systems, New York, pp. 1208–1210 (2006)
2. Milgrom, P.: Putting Auction Theory to Work. Cambridge University Press, Cambridge (2004)
3. Klemperer, P.: Auction Theory: A Guide to the Literature, Technical report, EconWPA (1999)

4. Monderer, D., Tennenholtz, M.: Optimal Auctions Revisited. Faculty of Industrial Engineering Technion – Israel Institute of Technology (1998)
5. Sandholm, T.: Distributed Rational Decision Making in Multiagent Systems. In: Weiss, G. (ed.), ch. 5. MIT Press, Cambridge (1999)
6. Lucking-Reiley, D.: Using Field Experiments to Test Equivalence between Auction Formats: Magic on the Internet. *The American Economic Review* 89(5), 1063–1080 (1999)
7. Bryan, D., Lucking-Reiley, D., Prasad, N., Reeves, D.: Pennies from eBay: the Determinants of Price in Online Auctions. *Econometric Society World Congress 2000 Contributed Papers* 1736, Econometric Society (2000)
8. Onur, I., Tomak, K.: Impact of ending rules in online auctions: the case of Yahoo.com. *Decis. Support Syst.* 42(3), 1835–1842 (2006)
9. Gregg, D.G., Walczak, S.: Auction Advisor: an agent-based online-auction decision support system. *Decis. Support Syst.* 41(2), 449–471 (2006)
10. Pardoe, D., Stone, P.: Developing Adaptive Auction Mechanisms. *SIGecom Exch.* 5(3), 1–10 (2005)
11. Cliff, D.: Evolution of Market Mechanism Through a Continuous Space of Auction-Types. In: *IEEE Congress on Evolutionary Computation on 2002*, Washington, DC, USA, pp. 2029–2034 (2002)
12. Wilson, S.W.: ZCS: A Zeroth Level Classifier System. *Evol. Comput.* 2(1), 1–18 (1994)
13. Zhang, J., Zhang, N., Chung, J.: Assisting Seller Pricing Strategy Selection for Electronic Auctions. In: *IEEE International Conference on E-Commerce Technology*, pp. 27–33. IEEE Computer Society, Washington (2004)
14. Vermorel, J., Mohry, M.: Multi-armed Bandit Algorithms and Empirical Evaluation. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) *ECML 2005. LNCS*, vol. 3720, pp. 437–448. Springer, Heidelberg (2005)

Managing Collaboration Using Agent Based Simulation

Utomo Sarjono Putro, Manahan Siallagan, Santi Novani, and Dhanan Sarwo Utomo

School of Business and Management, Institut Teknologi Bandung Jl. Ganesha No. 10,
Bandung 40132, Indonesia

utomo@sbm.itb.ac.id, manahan_siallagan@yahoo.com,
snovan8@yahoo.com, krigjsman@gmail.com

Abstract. The purpose of the present research is to identify, analyze and simulate dynamics of interaction and conflicts among agents using drama theory, and to apply it in Citarum river basin problem. To accomplish the purpose, we first model the process in terms of drama theory that is combined with emotional state model (PAD). One of the reasons why we adopt drama theory is that it primarily focuses on dilemma or paradox arising from rational goal seeking behavior. It also provides us with rigorous analytical and computational tools of conflict analysis. Then, we propose a simulation model to describe both of dilemma of conflict, *i.e.*, persuasion and rejection dilemma among the agents, and the dilemma of collaboration (trust dilemma) among the agents. Finally, we conduct agent-based simulation by using *SOARS (Spot Oriented Agent Role Simulator)* to obtain some fruitful suggestions for encouraging their collaboration.

Keywords: Agent based Simulation, Negotiation, Dilemma, Drama Theory, Emotion.

1 Introduction

Citarum River basin is a region in Java Island, Indonesia, with 6,080 km² area involving the three provinces, *i.e.*, West Java, Banten, and Jakarta. The annual precipitation depth is 3,000 mm/year in the mountain and 2,500 mm/year in lowland. Relative humidity is 80% and daily temperature is 25°C in the lowland and 18°C in the mountain. Citarum River is connected with 4 rivers to the west and 4 rivers to the east in West Java. In the past, Citarum river was clean, but now the condition had changed totally [17].

There are some factors which cause the problem, *i.e.*; illegal lodging and the population explosion in upper stream, and household waste in down stream. Agent in the Citarum problem has different interests and positions. There are several agents who participate in Citarum river basin, *i.e.* local people in downstream, local people in upstream, textile industries, environmentalist (green), regencies in upper stream and cities in down stream [19].

In negotiation process, agents may change their position and interests; accordingly, the situation is dynamic. There are some impediments (or dilemma) to achieve common position and trustworthy (*i.e.*, collaboration). Based on the previous research

[16], the effect of positive emotional state of agent is important to make negotiation. The results from simulation show that number of dilemma among agent could be reduced significantly or even have no dilemma.



Fig. 1. Citarum River Basin Region

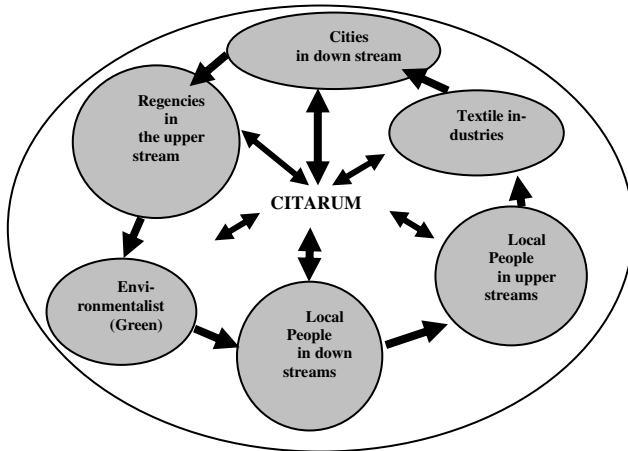


Fig. 2. Agents in Citarum River Basin Problem

In this research, we add a conceptual model for affiliation that is affiliate tendency. It was defined in terms of generalized positive social expectations and associated positive behaviors in social interactions with others.

Although dilemma of confrontation has been eliminated, dilemma of collaboration often still remains. Affiliate tendency is tendency to cooperate with other agent. It causes a new problem among the agent that is trust dilemma. How does each agent eliminate the trust dilemma? This paper will propose a simulation model of negotiation process involving dilemma of conflict and trust among the agents. Then we conduct agent-based simulation by using *SOARS (Spot Oriented Agent Role Simulator)* to obtain some fruitful suggestions for encouraging their collaboration.

2 Drama Theory in Citarum River Basin Problem

Drama Theory is a metaphor of dynamic confrontation process. This paper adopts the metaphor, and starts with the build up stage of interaction among agents in Citarum riverbasin problem. Common reference frame resulted from the stage is described by Fig. 3. The common reference frame consists of agents/participants, their options, positions (proposals), and threat.

OPTIONS OF PARTICIPANTS	THREAT	POSITIONS					
		USR	G	TI	DSP	USP	DSC
Up Stream Regencies (USR)						<	>
Stop deforestation	No	Yes	Yes	Yes/No	Yes	No	Yes
Green (G)				>	<	>	<
Protest	Yes	No	No	No	Yes/No	No	No
Textile Industries (TI)			>		>		>
Stop un-treatment waste disposal to river	No	Yes/No	Yes	No	Yes	Yes/No	Yes
Down Stream People (DSP)			<	<		<	<
Stop waste disposal to river	No	Yes/No	Yes	Yes/No	No	Yes/No	Yes
Up Stream People (USP)		>	>		>		>
Stop illegal lodging	No	Yes	Yes	Yes/No	Yes	No	Yes
Down Stream Cities (DSC)		>	<	>	<	>	
Strict penalties for illegal waste disposal to river	Yes	Yes/No	Yes	No	No	Yes/No	No
Maintenance River	No	Yes/No	Yes	Yes/No	Yes	Yes/No	No
Revenue sharing to Up Stream Regencies	No	Yes	Yes/No	Yes/No	Yes/No	Yes/No	No

Fig. 3. Common references frame in Citarum river basin problem

3 Agent Based Modeling in Drama Theory

The purpose of this paper is to identify, analyze and simulate dynamics of interaction and conflicts among the stakeholders (or agents) in the Citarum river negotiation process. They claim their strategies and interests as well as express emotion. To accomplish the purpose, we first model the process in terms of drama theory that is combined with emotional state model (PAD). One of the reasons why we adopt drama theory is that it primarily focuses on dilemma and also provides us with rigorous analytical and computational tools of conflict analysis. Then we conduct agent-based simulation by using SOARS. In this paper, we only discuss emotion model and temperament model (affiliative tendency). The detail of agent based modeling in drama theory could be find in the previous research [16].

3.1 Emotion Model and Affiliative Tendency

Emotion model that will be used in this paper is the development from emotional negotiation model PAD [9]. Emotional state model (PAD) involves three dimensions, *i.e.*, Pleasure (r_p), Arousal (r_a) and Dominance (r_d). During negotiation, a more pleasant agent tends to compromise with others. Each agent has the emotional state [9], *i.e.*:

$$Es_i = \{r_p, r_a, r_d\}; r_p, r_a, r_d \in (-1, 1) \tag{1}$$

Based on the previous paper [11], angry is coded by $\{-.51, .59, .25\}$, bored is $\{-.65, -.62, -.33\}$, curious is $\{.22, .62, -.01\}$, dignified is $\{.55, .22, .61\}$, elated is $\{.50, .42, .23\}$, hungry is $\{-.44, .14, -.21\}$, inhibited is $\{-.54, -.04, -.41\}$, loved is $\{.87, .54, -.18\}$, puzzled is $\{-.41, .48, -.33\}$, sleepy is $\{.20, -.70, -.44\}$, unconcerned is $\{-.13, -.41, .08\}$, and violent is $\{-.50, .62, .38\}$. Each agent i has the function of emotional state [9], which is:

$$Se_i(r_p, r_a, r_d) = r_p \cdot (1 + r_a) - r_d \tag{2}$$

For example, if an agent has emotional state defined as $\{-0.51, 0.59, 0.25\}$, then function of emotional state is $Se_i(r_p, r_a, r_d) = -0.51 \cdot (1 + 0.59) - 0.25 = -1.0609$.

Affiliate tendency was defined in terms of generalized positive social expectations and associated positive behaviors in social interactions with others. An individual’s emotional states are inferred from averages of his or her emotional states across representative samples of everyday situation. Thus, the previous paper [12] proposed that emotional traits could also be described in terms of the pleasure-displeasure, arousal-nonarousal, and dominance-submissiveness dimensions. Then, affiliate tendency scales were defined as follows:

$$Affiliation_{ij} = 0.46rp_{ij} + 0.24ra_{ij} + 0.3rd_{ij} \tag{3}$$

Within the present theoretical perspective, then, it is important to conceptualize and measure affiliate tendency as pure generalized interpersonal positive ness without either an inclination to want to dominate and control others or to be dominated and controlled by others.

3.2 Modeling to Eliminate Trust Dilemma

To eliminate trust dilemma in collaboration stage, we use affiliate tendency value. First, we calculate probability of cheating and punishing another agent who cheated.

$$Prob(i) = \frac{V_{\max} - aff_i}{V_{\max} - V_{\min}} \tag{4}$$

V_{\max} is maximum value of affiliate tendency (1.00); V_{\min} is minimum value of affiliate tendency (-1.00) and aff_i is affiliate tendency for agent i . The higher probability is the higher probability of the agent to punish another agent who cheated. In contrast, the lower probability is the higher probability of the agent to cheat.

Based on norm game [1], agent i will attempt to cheat the commitment if $Pr ob(i) > rand$, where $rand$ is random value which is generated from uniform distribution in range $[0,1]$. For each agent $j (j \neq i)$, he/she will attempt to punish agent i if $Pr ob(j) < rand$. If agent i cheat, then agent i 's *payoff* will increase by 1%, while the payoff of other agents will decrease by 1%. If agent i cheat and agent j punish, then payoff of agent i will decrease by 10%.

4 Simulation Using SOARS and Result

In order to simulate this problem, we use SOARS to describe the initial frame for Citarum river basin problem as seen in figure 3. There are so many dilemmas in the common frame. Based on the previous research [16], we can eliminate the dilemma of conflict. In this current paper, we assume that bargaining strategy of agent was same, that is $st_i = s \in S_i$. We make four experiments to look measure level of emotional state, so the agent could negotiate in order to reduce dilemma and to eliminate trust dilemma.

4.1 First Scenario

In this scenario, whole of agents have positive emotion. USR is having a strong desire to know about something, G is having strong feeling of deep affection for something,

Table 1. Parameters in Scenario 1

	<i>USR</i>	<i>G</i>	<i>TI</i>	<i>DSP</i>	<i>USP</i>	<i>DSC</i>
r_p	0.22	0.87	0.5	0.2	0.55	0.55
r_a	0.62	0.54	0.42	-0.7	0.22	0.22
r_d	-0.01	-0.18	0.23	-0.44	0.61	0.61
Se_i	0.3364	1.5198	0.48	0.5	0.0610	0.0610

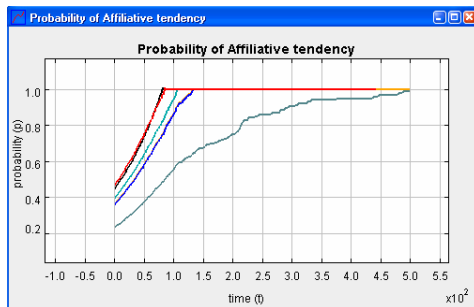


Fig. 4. Probability of Affiliate tendency in Scenario 1

TI is excitement, USP is calm, DSP is quiet and DSC is calm. In this scenario, we use the parameters as described in Table 1, and then the probability of affiliate tendency could be seen in figure 4.

From figure 4, we can see that the probability of affiliative tendency for each agent is close to 1, it means that no agent will cheat the commitment, because affiliate tendency is positive and the value is large. Each agent will avoid cheating because the other agent can punish him/her.

4.2 Second Scenario

In this scenario, number of agent who has positive emotion is more than number of agent who has negative emotion, and level of emotional state for agent with negative emotion is low. USR, USP, DSP, DSC and TI have positive emotion, meanwhile G has negative emotion. USR and DSC are excitement, USP and DSP have strong feeling of deep affection for something and TI is calm, G is not worried. In this scenario, we use the parameters as described in Table 2, and then the probability of affiliate tendency could be seen in figure 5.

Table 2. Parameters in Scenario 2

	<i>USR</i>	<i>G</i>	<i>TI</i>	<i>DSP</i>	<i>USP</i>	<i>DSC</i>
r_p	0.5	-0.13	0.55	0.87	0.87	0.5
r_a	0.42	-0.41	0.22	0.54	0.54	0.42
r_d	0.23	0.08	0.61	-0.18	-0.18	0.23
Se_i	0.48	-0.1567	0.061	1.5198	1.5198	0.48

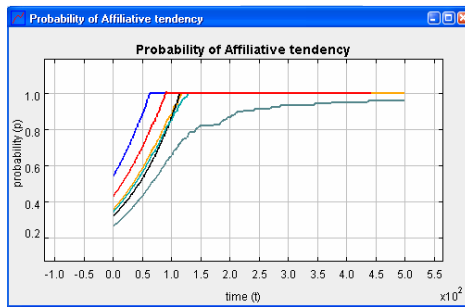


Fig. 5. Probability of Affiliate tendency in Scenario 2

From figure 5, we can see that the probability of affiliative tendency for each agent is close to 1, it means that no agent will cheat the commitment, because the affiliate tendency is positive and the value is large. Each agent will avoid cheating because the other agent can punish him/her.

4.3 Third Scenario

In this scenario, number of agent who has positive emotion is more than number of agent who has negative emotion, and level of emotional state for agent with negative emotion is moderate. USR, USP, DSP, DSC and TI have positive emotion, meanwhile G has negative emotion. USR and DSC are excitement, USP and DSP have strong feeling of deep affection for something and TI is calm, and G has a strong desire for something. In this scenario, we use the parameters as described in Table 3, and then the probability of the affiliate tendency could be seen in figure 6.

Table 3. Parameter in Scenario 3

	<i>USR</i>	<i>G</i>	<i>TI</i>	<i>DSP</i>	<i>USP</i>	<i>DSC</i>
r_p	0.5	-0.44	0.55	0.87	0.87	0.5
r_a	0.42	0.14	0.22	0.54	0.54	0.42
r_d	0.23	-0.21	0.61	-0.18	-0.18	0.23
Se_i	0.48	-0.2916	0.061	1.5198	1.5198	0.48

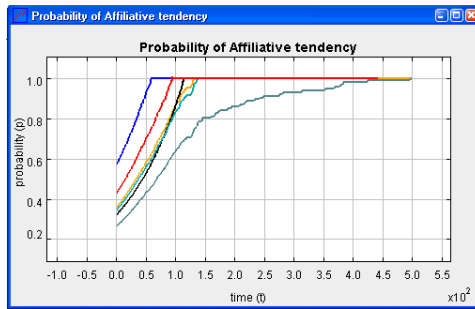


Fig. 6. Probability of Affiliative tendency in Scenario 3

From figure 6, we can see that the probability of affiliative tendency for each agent is close to 1, it means that all agents will keep their commitment, because affiliate tendency is positive and the value is large. Each agent will avoid cheating because the other agent can punish him/her.

4.4 Fourth Scenario

In this scenario, number of agents who have positive emotion is more than number of agents who have negative emotion, but level of emotional state for negative emotion is high. USR, USP, DSP, DSC and TI have positive emotion, meanwhile G has negative emotion. USR and DSC are excitement, USP and DSP have strong feeling of deep affection for something, TI is calm, and G is angry. In this scenario, we use the parameters as described in Table 4, and then the probability of affiliate tendency could be seen in figure 7.

Table 4. Parameter in Scenario 4

	<i>USR</i>	<i>G</i>	<i>TI</i>	<i>DSP</i>	<i>USP</i>	<i>DSC</i>
r_p	0.5	-0.5	0.55	0.87	0.87	0.5
r_a	0.42	0.59	0.22	0.54	0.54	0.42
r_d	0.23	0.25	0.61	-0.18	-0.18	0.23
Se_i	0.48	-1.045	0.061	1.5198	1.5198	0.48

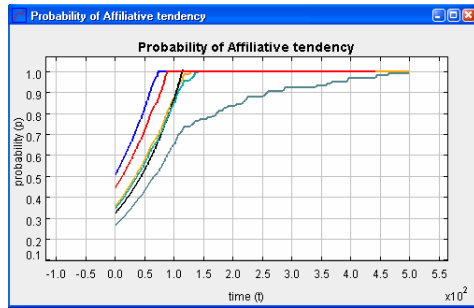


Fig. 7. Probability of Affiliate tendency in Scenario 4

From Figure 7, we can see that the probability of affiliative tendency for each agent is close to 1, it means that no agent will cheat the commitment, because affiliate tendency is positive and the value is large. Each agent will avoid cheating because the other agent can punish him/her.

5 Conclusion

From the results of simulation, we showed how the emotional states and affiliate tendency of agents significantly affect the negotiations process. In the collaboration stage, to maintain the commitment, each agent must be willing to punish the other agent who attempt to cheat. The effect of positive emotional state of agent and the affiliate tendency is important to maintain collaboration.

The results suggests that each agent should have a positive emotion and positive affiliate tendencies in order to achieve collaboration in the Citarum riverbasin problem.

References

1. Axelrod, R.: The Complexity of Cooperation. Agent-Based Models of Competition and Collaboration. Princeton University Press, Princeton (1997)
2. Bryant, J.: The Six dilemmas of Collaboration. John Wiley, Chichester (2003)
3. Barteneva, D., Lau, N., Reis, P.L.: Implementation of Emotional Behaviors in Multi Agent System Using Fuzzy Logic and Temperamental Decision Mechanism. In: Proceeding Fourth European Workshop on Multi Agent Systems (2006)

4. Bradley, M.M., Codispoti, M., Dean, S., dan Lang, P.J.: Emotion and Motivation II: Sex Differences in Picture Processing. *Emotion* 1(3), 300–319 (2001)
5. Howard, N.: Negotiation as Drama: How Games Become Dramatic. *International Negotiation* 1, 125–152 (1996)
6. Howard, N., Bennet, P., Bryant, J., Bradley, M.: Manifesto for a Theory of Drama and Irrational Choice. *Systems Practice* 6(4), 429–434 (1993)
7. Howard, N.: Drama Theory and its Relation to Game Theory: Part One. *Group Decision and Negotiation* 3, 187–206 (1994a)
8. Howard, N.: Drama Theory and its Relation to Game Theory: Part Two. *Group Decision and Negotiation* 3, 207–235 (1994b)
9. Jiang, H., Vidal, J.M., Huhns, M.N.: Incorporating Emotions into Automated Negotiation. University of South Carolina, Columbia (2004)
10. Maulana, F.: Menyelamatkan Hutan Tatar Sunda, *Kompas Online* 12 Mei (2004)
11. Mehrabian, A.: Questionnaire measures of affiliative tendency and sensitivity to rejection. *Psychological Reports* 38, 199–209 (1976)
12. Mehrabian, A.: Analysis of Affiliation Related Traits in Term of PAD Temperament Model. *The Journal of Psychology* 131, 101–117 (1997)
13. Putro, U.S.: Adaptive Learning of Hypergame Situations Using a Genetic Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics* 30(5) (2000)
14. Putro, U.S., et al.: Agent Based Modeling and Simulation of Knowledge Management. In: *Proceeding IFSR* (2005)
15. Putro, U.S., et al.: Drama Theory sebagai Model dari Dinamika Konflik dalam Permasalahan DAS Citarum. *Jurnal Manajemen Teknologi* 4(2) (2005)
16. Putro, U.S., et al.: Role of Emotion in Negotiation Process: An Application of Drama Theory in Citarum River Basin Problem. *International Society of System Science* (2007)

Using Agent-Based Simulation of Human Behavior to Reduce Evacuation Time

Arief Rahman, Ahmad Kamil Mahmood, and Etienne Schneider

Department of Computer and Information Science, Universiti Teknologi PETRONAS,
13750 Tronoh, Perak, Malaysia
rahmanarief@gmail.com, {kamilmh, dr_schneider}@petronas.com.my

Abstract. Human factors play a significant part in the time taken to evacuate due to an emergency. An agent-based simulation, using the Prometheus methodology (SEEP 1.5), has been developed to study the complex behavior of human (the ‘agents’) in high-rise building evacuations. In the case of hostel evacuations, simulation results show that pre-evacuation phase takes 60.4% of Total Evacuation Time (TET). The movement phase (including queuing time) only takes 39.6% of TET. From sensitivity analysis, it can be shown that a reduction in TET by 41.2% can be achieved by improving the recognition phase. Emergency exit signs have been used as smart agents. Modified Ant Colony Optimization (ACO) was used to determine the feasibility of the evacuation routes. Both wayfinding methods, the ‘familiarity of environment’, which is the most natural method, and the ACO method have been simulated and comparisons were made. In scenario 1, where there were no obstacles, both methods achieved the same TET. However, in scenario 2, where an obstacle was present, the TET for the ACO wayfinding method was 21.6% shorter than the one for the ‘familiarity’ wayfinding method.

Keywords: Evacuation planning, Prometheus methodology, multi-agent simulation, Ant Colony Optimization, and cognitive behavior.

1 Introduction

The owners of high-rise buildings must have a thorough plan for coping in the event of a disaster, such as fire, earthquake, bomb treat, etc. These plans must take into account the large number of occupants. Measures must be in place to prevent a situation from escalating. There must be adequate emergency facilities. Safe egress of occupants is of paramount importance [1].

The higher the number of occupants of high-rise building, the more attention should be given by building management to the safety regulations. Detailed calculations, based on a simulation or other modeling process, are required in order to appreciate the effect that building layout has on the evacuation process.

Even a single evacuation drill involving most of the occupants can be expensive. Furthermore, there is an inherent lack of realism, and, therefore, only limited confidence can be placed in any data gathered [2]. A computer based evacuation model has the potential of addressing these shortcomings.

The time taken to evacuate is the primary measure in assessing the effectiveness of an evacuation process [3]. Human, as the occupants of high-rise building, with varied behaviors and experiences, are the main actors in any evacuation process [4]. The complex human behaviors should be considered as the main factor in determining the time to evacuate. Some behaviors are potentially problematic and/or time wasting [5].

[6] has introduced an approach to model human cognitive behavior in the very beginning of a fire emergency. This human cognitive behavior model presents the pre-evacuation phase where most existing simulators has not completely presented. Moreover, [7] presents real evacuation drill data in four apartments where 50% of TET is lost during pre-evacuation phase. The author stated that occupants tend to ignore the fire alarm and are slow in responding to the emergency notification by continuing their activities. Unfortunately, different building complexity will have different characteristics of pre-evacuation time consumption.

The most complex aspect of people movement in an emergency condition is the approach to select the shortest way out from multi-exit ways in the high-rise building [8]. Other than the physical factors, there are some behavior-affected factors on making decision to choose the available routes. These are: familiarity of building environment (cognitive map) [8] [4]; interaction and cooperation within the group [4]; leadership factor among the occupants [9]; etc. Guidance or instruction is necessary and important for occupants in panic situations. Exit signs are one type of guidance to find the alternative routes but it is only a static label. A leader among a group of evacuees can also offer guidance and the response from the followers will be higher than with using an exit sign [10]. Unfortunately, it is not simple to find the leader in every occupant group during a panic situation. Most occupants tend to act more individually and lack the leadership skill to guide others.

The human cognitive behavior model built in the simulation presents the study over lengthy time periods during the pre-movement phase. In order to improve the wayfinding method, this paper also presents a comparison between the ACO wayfinding method and the 'familiarity of environment' wayfinding method.

2 Related Works

Since the last four decades, there has been a tremendous growth in computer simulation and modeling of evacuation planning in high rise buildings. Most of the existing evacuation simulators, such as SGEM [1], SIMULEX [14], EVACNET [23], and building-EXODUS [25], specifies the maximum number of saved evacuees and minimum TET as the two main parameters of an evacuation process.

The development in the study of human behavior in pre-evacuation is not as fast as the development of evacuation simulators. With agent-based system, [4] has developed a prototype of model for non-adaptive crowd behavior, including some behavior in pre-evacuation phase. A cognitive behavior model has been proposed by [6] with a probabilistic decision tree model to represent some human behaviors during pre-evacuation. Unfortunately, this cognitive behavior model has not been developed into computer simulation.

Multi-agent based systems have been found sufficient to represent the complex human behavior and decision making process. [4] has developed a multi-agent simulation as a basic scheme in evacuation planning where some human behaviors were

attached to the agent. In order to study the leadership contribution to the evacuation process, [10] has introduced a leader in the simulator. The multi-agent system has been modified to represent the interaction among leaders and other occupants. Another application of multi-agents is presented by [9] to simulate the leader behavior during evacuation. An agent leader has the capability to lead the other occupants to perform high-level wayfinding by obtaining the cognitive map of a building. The leadership aspect and other human responses during an evacuation process have also been represented by [12] using an agent in virtual participatory simulation.

3 Evacuation Phases

There are two main phases in an evacuation process i.e., pre-evacuation phase/response and movement phase/evacuation [13] [14]. The main parameter of emergency evacuation is TET [24]. TET is formed by three time components namely response time (t_{rest}) or pre-evacuation time (PeT), moving time (MT) and waiting time (WT).

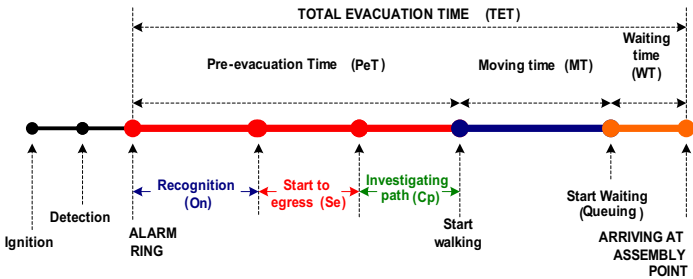


Fig. 1. Evacuation time line, where TET starts from the alarm ring event

4 Human Behaviors in Evacuation Process

4.1 Panic

Life threatening situation in the event of fire or any other disaster can be a triggered event of panic, which could possibly lead to accidents or casualties of human lives because of crushing or trampling [15]. When some clues are received followed by a perception of a dangerous situation, people often act irrationally unless they have a strong positive social personality such as leadership capability [4]. People who fall into panic, usually lose their judgement and may cause uncontrolled evacuation process. In such situations, the availability of guidance will reduce the panic behavior.

In a panic situation, people move with dynamic movement where the velocity is influenced by some particular force. The dynamic movement is determined by the acceleration of movement. According to [15], there are two forces that influence the movement, i.e. socio-psychological and physical forces. The dynamic movement represents the direct interaction between human and the physical object in the building. The acceleration equation (1) describes the change of people's velocity with time t .

$$m_i \frac{dv_i}{dt} = m_i \frac{v_i^0(t).e_i^0(t) - v_i(t)}{\tau_i} + \sum_{j(\neq i)} f_{ij} + \sum_W f_{iW} \quad (1)$$

while the change of position $r_i(t)$ is given by the velocity $v_i(t)=dr_i/dt$.

4.2 Wayfinding

Quoted from [9], “*Wayfinding is the process of determining and following a route to some destination*”. This process needs the cognitive component of navigation and building knowledge to determine the route from initial position to targeted position. [4] has classified the individual decision making process during an evacuation into three basic conventions, those who follows instinct, follows experience and bounded rationality.

Following instinct is the most primitive decision by people in making instantaneous and quick response [4]. Naturally, human are able to retrieve their past experiences and follow their habitual activities or repetitive events in making decision. During an evacuation process, the previous experiences of the occupants have a significant effect on their responses to the emergency situation in the building [4]. The familiarity of building environment, some knowledge related to safety procedures and evacuation drill experiences are some experiences that directly influence their decisions during an emergency situation.

4.3 Ignoring Immediate Leaving

To date, few studies in existing literatures have observed and analyzed human behavior during pre-evacuation phase. Whereas, previous observations of actual evacuation drills show that some behaviors such as saving valuable items, saving important documents have caused time wasting during pre-evacuation phase. According to OSHA standard procedure, once the emergency status is announced, all the occupants excluding safety officer or floor warden must evacuate immediately from the building. On the contrary, some previous studies presented by [1], [7] and [11] and also our evacuation survey result show that most occupants will not heed to the emergency notification. This behavior is termed as ignoring immediate leaving.

5 Agent-Based Modeling

5.1 Prometheus Methodology

Prometheus, a methodology to construct multi-agent systems and the detail components of agents [17], can be classified as a top-down approach. The Prometheus methodology is a refined system that includes from system objective to detailed planning of each agent as a systematic hierarchy breakdown. There are three main phases in Prometheus i.e. system specification, architectural design and detailed design [17].

Even though there have been several tools and methodologies introduced for agent systems development, it is not recommended to select one particular methodology as being the best. In fact, the type of problem and scope of application should be considered before selecting the most suitable methodology [18]. Prometheus has a systematic phase

to build intelligent agents [19]. From the scale of details point of view, Prometheus has provided a complete phase with detailed specification and it has a clear concept to represent agent with high autonomy and mental attitude [18]. From some practical parameters, i.e. clear notation, ease of learning, ease of use, adaptability, traceability, consistency, and refinement, Prometheus methodology has fulfilled those standard criteria [19].

5.2 Model Development

Prometheus design tool (PDT) version 3.1 [20] has been applied to ensure that the development of evacuation system are appropriate with Prometheus methodology's role.

5.2.1 System Specification

The simulation of emergency evacuation in a multi-level building has some complex aspects that need to be considered such as occupant with unique behaviors. [4]. Therefore, it is necessary to define the specification of the evacuation system so that the scope of study becomes more focused and directed. Fig. 2 shows an overview of a system specification.

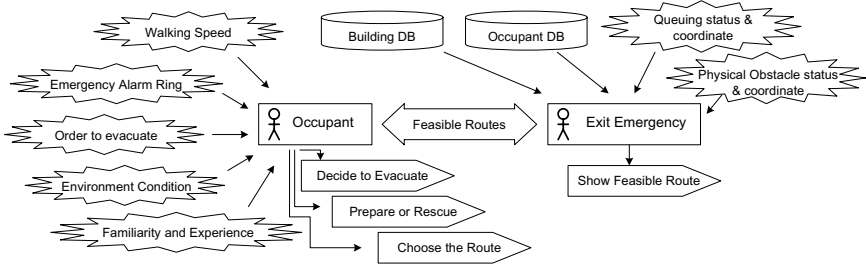


Fig. 2. An overview of an evacuation system

Goal Overview. There are 2 main goals to be achieved in the evacuation process [3], i.e., *minimizing TET* (Goal #1) and *maximizing the number of evacuees* (Goal #2).

Functionalities. There are four defined functionalities to manage and operate the interaction during simulation in order to achieve the goals. The first function i.e., *calculate number of people in queue* has the function to control the length of queue and calculate the utilization of building spaces. Functionality #2, *identify physical obstacle* has the function to identify the location of fire or other damages in the building in order to avoid any potential accidents. *Determine the feasible route*; functionality #3 is an important function to maximize the number of saved evacuees by calculating the most feasible route. Operating and managing people's response to alarm warning system is the function of *order people to evacuate* (functionality #4).

Scenarios. The first scenario, *leaving immediately*, has been developed to describe some actions that will be taken by occupants when the emergency alarm rings. This scenario is triggered by the emergency alarm and is influenced by the familiarity of environment or experience of occupants. The second scenario, *finding an obstacle*, has been created to provide some actions taken by the emergency system when an

obstacle appeared in the building. The system will obtain some information from sensors in the building to identify the obstacle such as queuing obstacle, physical obstacle and environmental conditions.

5.2.2 Architectural Design

A complex design of multi-agent system specification has been incorporated into the architectural design phase. Occupants are defined as the agents in the evacuation process, who are capable to response, react, interact and perhaps reject each other. The emergency exit sign has been modified to be a dynamic agent, which is able to determine the feasible route. There are 7 percepts that have been introduced into the evacuation system and 6 have been set into simulation as depicted in fig.2.

The interaction between agent staircase, corridor/hall and agent emergency exit has been provided by protocol *physical obstacle status* (protocol #1). The message of *obstacle status* is updated continually to agent emergency exit. Protocol #2 (*location of occupant*) also shows the interaction between agent staircase, agent corridor/hall, and agent emergency exit sign. A message and coordinate of each occupant is provided and updated by agent staircase and agent corridor/hall. Protocol #3, *feasible route*, connects the agent emergency exit and the agent occupant. Agent emergency exit sign periodically sends the message of feasible route to agent occupant.

5.2.3 Detailed Design

Detail structures and components of agent are provided in this part of the model.

Agent Emergency Exit Sign. Feasible route determination in evacuation planning performed by agent emergency exit sign will support the main objective of the simulation i.e. getting minimum TET. When the simulation clock starts, this agent will receive some information related to the location of each occupant and obstacles from other agents.

The capability of feasible route determination is built with the Ant Colony Algorithm [21], which enables the determination of the feasible route by calculating the shortest route and avoiding potential obstacles that appear in the building. The information concerning the feasible route is transmitted to agent occupants by sending a message to all exit signs in the building.

The Ant Colony Algorithm as a plan inside the feasible route determination capability is designed to obtain some input regarding occupant location, distance to assembly point and building specification. In order to apply ACO on agent emergency exit sign, a new factor (ω_{ij}) is added by considering some physical obstacles in the building to be evacuated such as fire location, damaged facilities, bottleneck problem and obstacle on the exit corridor [22]. A route with low probability, where a physical obstacle has appeared, should not be chosen. A modified transitional probability rule, as given in equation (2), determines the next route that will be chosen by an ant.

$$P_{ij}^k(t) = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta \cdot [\omega]^\lambda}{\sum_{k \in \text{allowed } k} [\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta \cdot [\omega]^\lambda} \quad (2)$$

Where α , β , λ are parameters that control the relative importance of those 3 variables of transitional probability rule.

Algorithm 1. Procedure 'Ant Colony Algorithm'

```

Connect to Occupant DB
Connect to Building DB
Read initial location of occupant
Read initial Obstacle status
Read walking speed
Calculate distance between nodes
Set number of cycle
Set number of ants=total number of occupants
Set initial pheromone, i to j
Set initial probability function, i to j
For i=1 to number of occupant
    Place ants to location of occupant
    Set pheromone, i to j
    Update pheromone, i to j
Update obstacle status
Calculate probability function, i to j
    Update probability function, i to j
Choose appropriate value of probability function
Move ant to next node
If ant arrive at assembly point then
    Update the distance of trip
    Update number of circle
    If distance of trip = shortest route then
        Set shortest route = distance of trip
    End if
End if
Next i
Send feasible route through protocol

```

Agent Occupant. As the main actor in the simulation, the proactive behavior of agent occupant is represented by the agent's ability to response to any changes within the environment. Occupant movement control is built inside the *response and move* capability.

The *response and move* capability is divided into *pre-evacuate response* and *movement and interaction* capabilities. Certain roles that determine the response by agent occupant, has been built inside the *pre-evacuate response* capability by applying certain probabilistic values. According to [6], a method to assess human cognitive behavior in evacuation is applied to the system and the Single Value Network (SVN) to start egress motion is attached to the simulation.

The *movement and interaction* capability determines the movement speed of the occupant and represents the interaction among the occupants. The acceleration of people movement in panic [15] is applied in this capability.

5.3 Simulation Setup

One of the student's hostels at our university has been chosen as a case problem for simulation. This building has 4 stories, each floor has 4 blocks of rooms, each block has 6 rooms (excluding 1 bathroom and 1 kitchen in every block), and each room has 2 occupants. In this simulation, each person is defined as male with an average height of 160 cm and walking speed of 1.8 m/s. The average body size is 0.5m x 0.5m

(a square). Other physical attributes can be adjusted by modifying some input on SEEP 1.5. The maximum number of occupants involved is 180 occupants and this number represents the actual number of occupants in the hostel.

5.4 Validations

A simulation model is a means to represent the problems in real world into conceptual model. SEEP 1.5 is validated by using the black-box validation process where actual walking time was compared with the walking time produced by the simulator. A second validation was also carried out using EVACNET 4 [23]. A comparison between SEEP 1.5 and EVACNET 4 shows that there are no significant differences between the two. The walking time produced by SEEP 1.5 almost matches the real walking time. SEEP 1.5 performed the simulation as well as EVACNET 4.

6 Simulation Results

6.1 Pre-evacuation Study

An evacuation survey was conducted to obtain some information related to possible actions by the occupants upon hearing the emergency alarm notification. The probability values of pre-evacuation actions on SVN (Single Value Networks) model are defined as follows: leaving probability (p_{on}) = 0.278, preparing probability (p_{se}) = 0.647, and choosing probability (p_{cp}) = 0.471. Those probability numbers were applied on SEEP 1.5 to generate the pre-evacuation time of each occupant.

In most cases, the pre-evacuation time tends to be skewed because some occupants may take longer time to response while others may respond faster. In the case of hostel evacuation, the simulated pre-evacuation time is in good fit with the Weibull distribution.

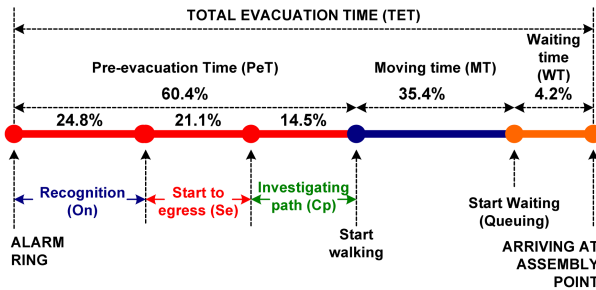


Fig. 3. Time proportion for each phase in hostel evacuation

As seen in fig. 3, TET is divided into 3 components of time. On average, time to prepare or pre-evacuation time reaches more than 50% of TET, while time to move takes around 40% of TET and time to queue only takes around 10% of TET. Compared with the experimental results obtained by [7], this simulation result shows a similar characteristic of time consumption. ‘Ignoring the emergency alarm notification’ (recognition phase) takes the longest time during pre-evacuation or 24.8% of

TET. Occupants also required a long period of time (21.1% of TET) to save their valuable items before leaving the building and ‘Investigating the path’ takes 14.5% of TET.

6.2 ACO Wayfinding

In the hostel simulation, emergency exit sign has been modified as a smart agent, where ACO was embedded in the emergency exit agent. ACO functions to determine the feasible route based on some percepts received from the building based on a real time situation.

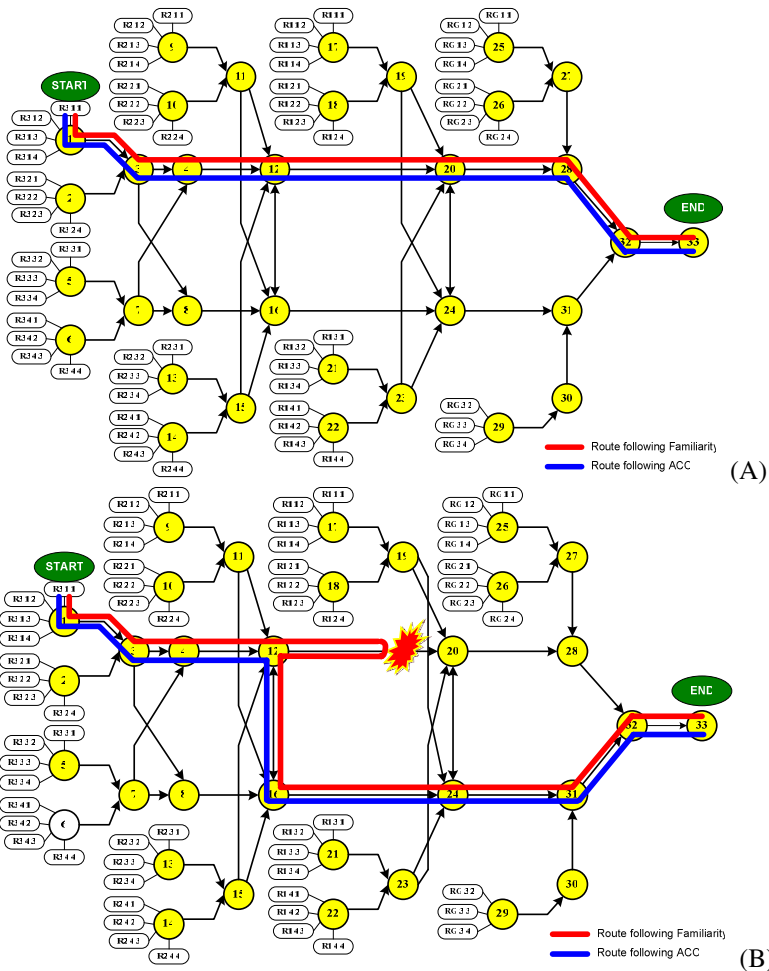


Fig. 4. A comparison between ‘familiarity of environment’ wayfinding method and ACO wayfinding method, without obstacle (A) and with obstacle (B)

6.2.1 Scenario 1 (Wayfinding Methods Comparison without Obstacle)

The first wayfinding method, familiarity of environment, generally follows the occupant's routine and the most common choice is to take the staircase and exits, nearest to their current position. Usually, the taken route follows the familiarity of environment in a normal situation and is straight forward. This route should be taken because it is the shortest route formed by their daily experience.

The hypothesis of this scenario tends to compare the performance of those two different wayfinding methods. The total movement time is measured as the main comparison parameters. From the simulation result and based on t-test result, it can be interpreted that there is no significant different in movement time perform by both wayfinding. Figure 4. A shows that in the absence of obstacles, the simulation results give the same route either with ACO or familiarity of environment methods.

6.2.2 Scenario 2 (Wayfinding Methods Comparison with an Obstacle)

As described in fig. 4, the obstacle has impeded the path through the left staircase L2 - L1 (level 2 to level 1). That obstacle has forced the occupants to choose that route and turned back through another staircase on the right side.

With familiarity wayfinding method, some occupants were trapped on the left staircase L2-L1 and had to turn back to the right side, thus creating a bidirectional crowd flow on the right staircase. Using ACO wayfinding method, most of the occupants were able to avoid the obstacle and selected the route through right staircase L2-L1 directly. The ACO has considered the position of the obstacle and hence avoided the blocked route.

Based on t-test result, total movement time due to familiarity of environment wayfinding method is significantly different from the total movement time guided by ACO wayfinding method. In the case of hostel evacuation, the average clearance time taken by familiarity wayfinding method is **21.6%** longer than the average clearance time taken by ACO wayfinding method.

7 Discussions

Based on our evacuation survey and previous observation reports, the occupants are usually slow in responding to emergency alarm notification. Some completely ignore the signal and continue their activities even after the alarm has rung. As stated by [16], there are three possible reasons why people ignore the alarm signal, i.e.: failure to recognize the alarm signal as an emergency alarm, disbelief of the emergency system because of occasional nuisance alarms, and unable to hear the emergency warning through the alarm.

There are some examples of nuisance alarm, such as false alarm, alarm in the event of evacuation drill, and test alarm. Since many nuisance alarms sounded in the building, the occupants might assume that the alarm signal is a false alarm. Our survey results show that 30% of sounded alarm ring in the event of evacuation drill and 25% sounded alarm ring because of false alarm. This condition reduces the people's confidence and trust level of the emergency system. Therefore, a high precision emergency detection device should be installed by building management in order to prevent nuisance alarm.

Leaving probability and choosing probability sensitivity were studied to measure their effect on TET by setting a constant preparing probability. With reference to hostel evacuation, if the building management had been able to notify all occupants for immediate evacuation once the alarm rings (leaving probability = 1 or recognition time ≈ 0), then there will be an opportunity to reduce the TET by $\pm 41.2\%$ (compared to normal conditions where leaving probability = 0.28).

In the above experiments, the two wayfinding methods have been compared and the simulation results show that ACO wayfinding has a shorter clearance time than familiarity of environment wayfinding. When physical obstacle(s) appeared in the building, untrained occupants with less experience and inability to locate the position of the physical obstacle might be trapped at the impeded location. Based on the most possible conditions and also from simulation scenario 2, it is recommended that evacuation plan must be prepared to handle the worst condition inside a high-rise building, emphasizing especially on the preparedness for dynamic evacuation.

Achieving minimum TET is associated with successful evacuation process, which means that more people can be evacuated safely. In normal situation, the path formed by the daily routine movement forms the shortest exit of the building. However, in an actual emergency situation, when obstacles must be considered, the shortest route does not necessarily mean the safest route anymore. This situation is clearly shown in the simulated scenario 2. Their familiarity of the environment does not provide them the ability to either identify the status or locate the coordinate of the obstacle.

8 Conclusions

SEEP 1.5 has been used successfully to model human cognitive behaviors for estimating the pre-evacuation time. The simulation results show that pre-evacuation phase consumed 60.4% of TET. By conducting sensitivity analysis, TET can be reduced by 41% by eliminating the recognition time in pre-evacuation phase.

As the natural and the most common wayfinding method, familiarity wayfinding method was able to determine the shortest evacuation route where the obstacle existence was denied. However, when an obstacle appeared in the building, the routine route cannot be relied on any longer. In fact, people need guidance in panic situation and safety must be the primary objective in an evacuation process.

Modified ACO embedded on agent emergency exit sign was able to determine the evacuation route by recognizing the obstacle status in the building. The simulation results of a hostel evacuation show that when an obstacle appears, the ACO wayfinding method performs 21.6% faster than the familiarity wayfinding method. The comparison between the two methods shows that a local based decision does not assure the occupant to take the feasible route during an emergency evacuation. Considering safety, the term "feasible route" is more appropriate to be used rather than the term "shortest route" because the shortest route might not always mean the safest feasible route.

In order to expand the scope of this simulator, other more challenging scenarios can be applied to include more extreme building environment such as a nuclear power plant, a crowded sports stadium or skyscrapers. The prediction of important evacuation factors for such buildings can be used for better evacuation planning and helps protect more lives.

References

1. Lo, S., Fang, Z., Zhi, G., Yuen, K.: A computer simulation model of emergency egress for space planners. *Journal of Facilities (Emerald)* 20, 266–270 (2002)
2. Johnson, C.: Lessons from the evacuation of the WTC, September 11th 2001 for the development of computer-based simulations. *Cognition, Technology & Work Journal* 7, 214–240 (2005)
3. Gwynne, S., Galea, E.R., Owen, M., Lawrence, P.J., Filippidis, L.: A review of the methodologies used in computer simulation of evacuation from the built environment. *Journal of Building and Environment* 34, 741–749 (1999)
4. Pan, X., Han, C.S., Dauber, K., Law, K.H.: Human and social behavior in computational modeling and analysis of egress. *Journal of automation in construction* 15, 448–461 (2006)
5. Purser, D., Bensilum, M.: Quantification of behaviour for engineering design standards and escape time calculations 38, 157–182 (2001)
6. Pires, T.: An approach for modeling human cognitive behavior in evacuation models. *Fire safety journal* 40, 177–189 (2005)
7. Proulx, G.: Evacuation time and movement in apartment buildings. *Fire Safety Journal* 24, 229–246 (1995)
8. Lo, S.M., Huang, H.C., Wang, P., Yuen, K.K.: A game theory based exit selection model for evacuation. *Fire Safety Journal* 41, 364–369 (2006)
9. Pelechano, N., Badler, N.I.: Modelling crowd and trained leader behavior during building evacuation, pp. 80–85. IEEE Computer Society, Los Alamitos (2006)
10. Murakami, Y., Minami, K., Kawasoe, T., Ishida, T.: Multi-agent simulation for crisis management. In: *Proceeding of the IEEE workshop on knowledge media networking*. IEEE, Los Alamitos (2002)
11. Olsson, P., Regan, M.: A comparison between actual and predicted evacuation times 38, 138–145 (2001)
12. Sugimoto, Y.: Modeling action rules through participatory simulation in virtual space. Master Thesis, Kyoto University, Japan (2005)
13. Chow, W.: "Waiting time" for evacuation in crowded areas. *Journal of Building and Management Journal* 42, 3757–3761 (2007)
14. Thomson, P.A., Marchant, E.W.: A computer model for the evacuation of large building populations. *Fire Safety Journal* 24, 131–148 (1995)
15. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic 407, 487–490 (2000)
16. Proulx, G.: Evacuation time and movement in apartment buildings. *Fire Safety Journal* 24, 229–246 (1995)
17. Padgham, L., Winikoff, M.: Developing intelligent agent systems, a practical guide. In: West Sussex. John Wiley & Sons Ltd., Chichester (2004)
18. Tran, Q., Low, G.: Comparison of ten agent-oriented methodologies. In: Henderson, B., Giorgini, P. (eds.) *Agent-Oriented Methodologies*. Idea group publishing, Hershey (2005)
19. Al-Hashel, E., Balachandran, B., Sharma, D.: A comparison of three agent-oriented software development methodologies: ROADMAP, Prometheus, and MaSE. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) *KES 2007, Part III. LNCS (LNAI)*, vol. 4694, pp. 909–916. Springer, Heidelberg (2007)
20. RMIT Intelligent Agents Group, <http://www.cs.rmit.edu.au/agents/pdt/>
21. Dorigo, M., Maniezzo, V., Colomi, A.: Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics* 26(1) (1996)

22. Rahman, A., Kamil, A.: Feasible route determination using ant colony algorithm in evacuation planning. In: ScoRed 2007. IEEE, Malaysia (2007)
23. Kisko, T.: EVACNET4 - A program to determine the minimum building evacuation time. University of Florida (1999), <http://www.ise.ufl.edu/kisko/files/evacnet/>
24. Proulx, G.: Evacuation time and movement in apartment buildings. *Fire Safety Journal* 24, 229–246 (1995)
25. Gwynne, S., Galea, E., Owen, M., Lawrence, P.J., Filippidis, L.: A systematic comparison of building EXODUS predictions with experimental data from the Stapelfeldt trials and the Milburn House evacuation. *Applied mathematical modelling* 29, 818–851 (2005)

Participatory Simulation Platform Using Network Games

Shoichi Sawada¹, Hiromitsu Hattori¹, Marika Odagaki², Kengo Nakajima²,
and Toru Ishida¹

¹ Graduate School of Informatics, Kyoto University,
Yoshida-Honmachi Sakyo-ku Kyoto 606-8501 Japan
shoichi@ai.soc.i.kyoto-u.ac.jp

² Community Engine Inc., 4-31-8 Yoyogi Shibuya-ku Tokyo 151-0053 Japan

Abstract. In this paper, we develop a novel participatory simulation platform, called *gumonji/Q*, by integrating scenario description language *Q* and network game *gumonji*. In a participatory simulation, humans and software-agents coexist in a shared virtual space and jointly perform simulations. To observe practical behaviors of humans, a participatory simulation platform must be able to provide reasonable simulated experience for humans to let them behave as they do in the real-world. *gumonji/Q* makes it possible to design diverse interaction protocols based on *Q*'s scenario description ability. Under the “game-quality” graphics provided by *gumonji*, humans and agents can interact with their surrounding environment, which means they can affect the environment and receive feedback from the environment. Since *gumonji/Q* inherits *gumonji*'s features as a network game, users are more enticed to participate in simulations since simulations on *gumonji/Q* seems more enjoyable than normal simulations. We show an example of how to obtain human behavior models through a simulation on *gumonji/Q*.

Keywords: Multiagent Simulation, Participatory Modeling, Participatory Simulation, Gaming, Networked Simulator.

1 Introduction

Social simulations are becoming popular for designing socially embedded systems. Multiagent-based simulation (MAS), which can reproduce complex systems (*e.g.* human society) based on a bottom-up approach, is considered as one of the promising way for conducting social simulation [1,2]. In order to achieve MAS, computation models for reproducing human's behaviors are required. We have focused on participatory modeling as an influential methodology for obtaining practical human behavior models [3]. Although there are several ways for conducting participatory modeling [4,5], we are trying to develop a modeling methodology using participatory multiagent-based simulation (PMAS). This is because PMAS is suitable to provide better real-world experience and make it possible to achieve massively simulations with a number of users [6].

In PMAS, humans and software-agents jointly perform simulations, so that there are technical issues for developing a PMAS platform which can provide reasonable simulated experience. During simulations, humans interact with agents and the surrounding environment. In order to let humans behave practically, they are able to interact with agents based on social interaction protocols. Humans also are able to affect the surrounding environment and receive feedback. Especially, it is important that humans can visually recognize the current state to let them make a decision in a natural way.

Following the above discussion, we intend to develop a PMAS platform for obtaining practical human behavior models. We develop a novel networked participatory simulation platform, called *gumonji/Q*, by integrating scenario description language *Q* and network game *gumonji*¹. *Q* is a scenario description language that enables us to describe complex social interactions between agents and their surrounding world (humans, other agents, simulation environment, etc.) [7]. *gumonji* is a network game which offers playing fields for virtual life. Users can take a large variety of activities via characters on that field as though they are in the real-world. They also can change their surrounding environment and visually recognize any changes through game-quality graphics. *gumonji/Q*, thus, is able to realize a participatory simulation environment where “human-agent” and “human(agent)-environment” relationships could be reasonably achieved to reproduce and understand a large variety of social phenomena.

The remainder of the paper is organized as follows. First, we show the background of our research; related works and two primary software (*Q* and *gumonji*). Second, we present new participatory simulation platform *gumonji/Q*. Then, we show an example of participatory modeling on *gumonji/Q*. Finally, concluding remarks are given in the final section.

2 Background

2.1 Related Works

FreeWalk is a platform for constructing virtual collaborative events in which agents and humans can socially interact with each other in a virtual space [8]. An agent is controlled through the platform’s API. A human participant enters the virtual space as an avatar, which he/she controls through the UI devices connected to the platform. The primary objectives of FreeWalk is to realize a virtual space for communication and collaboration between humans and agents. However, while humans and agents can take diverse social interaction on FreeWalk, it does not simulate the environment in detail, which means humans cannot affect their surrounding environment and receive feedback. For providing better real-world experience, we need an interactive simulation environment.

CORMAS can be used to build simulation models of coordination modes between individuals and groups who jointly exploit the resources [9]. In CORMAS, users can define the diffusion of environmental changes, and agent’s behaviors

¹ <http://www.gumonji.net/>

followed by the surrounding environment. The computational model behind CORMAS simulations is a cellular automaton. A natural environment is modeled as a two dimensional mesh, and the diffusion between neighboring cells is calculated at each unit time. CORMAS is useful to describe interactions between natural environment and humans. However, while FreeWalk emphasizes graphics functions, CORMAS just shows abstract graphics, so that it does not provide enough visual information and it is hard for humans to behave like in the real-world on the environment described as two dimensional mesh.

2.2 Scenario Description Language *Q*

Q is a scenario description language for multiagent systems that allows us to define how agents are expected to interact with its environment involving humans and other agents [7]. *Q* is suitable for describing complex social interactions [10,11]. *Q* scenarios foster the emergence of dialogs between agent designers (computing professionals) and application designers (scenario writers) [12]. The computational model behind a *Q* scenario is an extended finite state automaton, which is commonly used for describing communication protocols. By using *Q*, users can directly create scenario descriptions from extended finite state automata. *Q*'s language functionality is summarized as follows:

- Cues and Actions

An event that triggers interaction is called a cue. Cues are used to request agents to observe their environment. Cues keep on waiting for the event specified until the observation is completed successfully. Comparable to cues, actions are used to request agents to change their environment.

- Scenarios

Guarded commands are introduced for the situation wherein we need to observe multiple cues simultaneously. A guarded command combines cues and actions. After one of the cues becomes true, the corresponding action is performed. A scenario is used for describing protocols in the form of an extended finite state machine, where each state is defined as a guarded command. Scenarios can be called from other scenarios.

- Agents and Avatars

Agents, avatars and a crowd of agents can be defined. An agent is defined by a scenario that specifies what the agent is to do. Even if a crowd of agents executes the same scenario, the agents exhibit different actions as they interact with their local environment (including other agents and humans). Avatars controlled by humans do not require any scenario. However, avatars can have scenarios if it is necessary to constrain their behavior.

2.3 Network Game *gumonji*

gumonji is a network game with the function of environmental simulation that is developed and released by Community Engine Inc. Figure 1 shows a snapshot of the virtual space of *gumonji*. In *gumonji*, animals and plants exist in a virtual

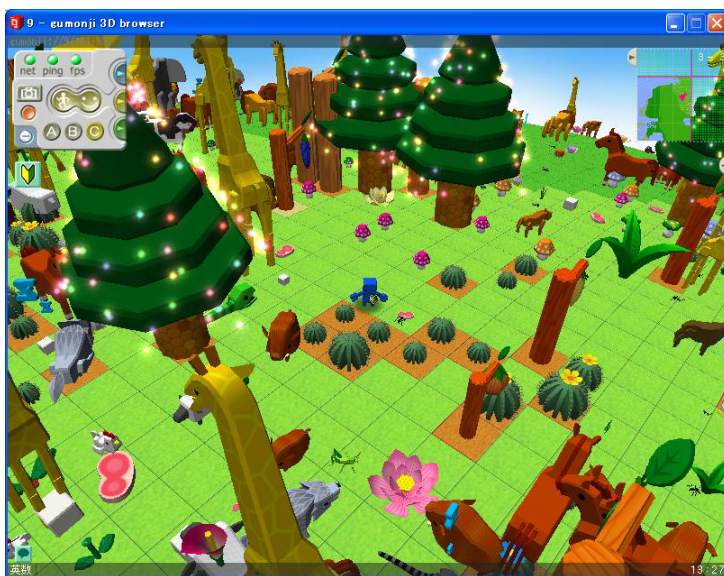


Fig. 1. A Screenshot of gumonji

space, and the atmosphere and water circulate according to physical law. Users participate in the simulation environment by operating characters (avatars²) in the virtual space, and can manipulate living things through actions of characters. The influence given to the living thing spreads widely time passes, and the environment change is displayed in 3D virtual space. Many users can participate in the same environment, and communicate with other users through actions and chats of avatars.

In *gumonji*, all users have own simulation environment on individual computers and participate in other users' simulation environment through P2P network. By using P2P network, users can move freely across multiple simulation environments and participate anytime and anywhere.

Since *gumonji* is a network game where many users participate and communicate with other users, actors on the environment are all characters operated by users. Therefore, *gumonji* does not have the function to construct and run autonomous agents. For the purpose of participatory simulation, it is necessary to add the function to describe diverse interaction pattern for agents.

3 Networked Simulator *gumonji/Q*

For achieving participatory simulation, we think two types of interaction should be available, *i.e.* interaction between entities (humans, agents, etc.) and interaction between an entity and environment. As we mentioned in the previous

² In this paper, we call a human-controlled character "avatar" in order to distinguish human-controlled character(agent) and *Q*-controlled agent.

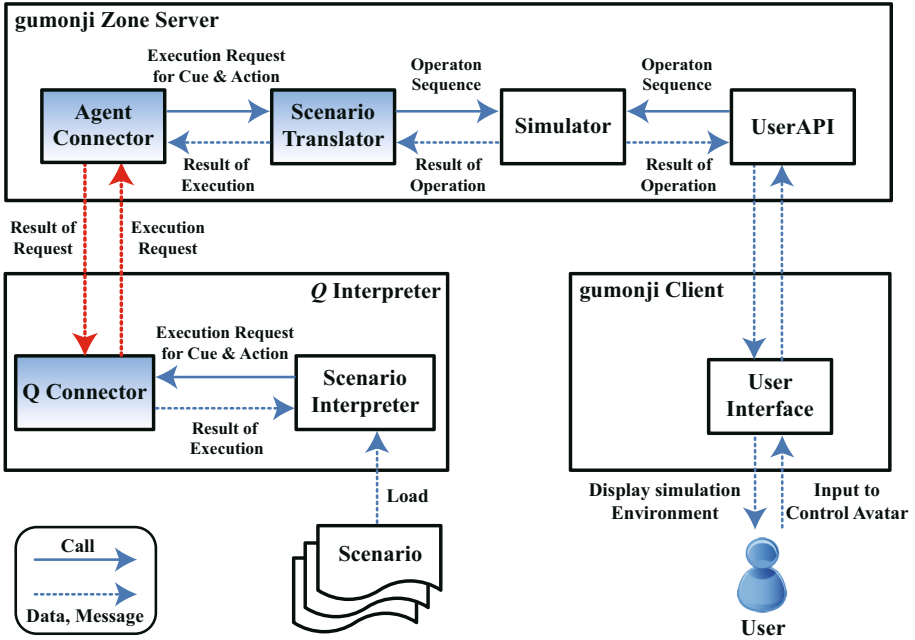


Fig. 2. An Architecture of gumonji/Q

section, for the design of interaction between entities, a scenario description language *Q* is useful, and we can obtain interactive simulation environment by using *gumonji*. In this section, we show how to connect these two software to construct a simulation platform *gumonji/Q*.

3.1 An Architecture of gumonji/Q

Figure 2 shows an architecture of *gumonji/Q*. Colored boxes are newly implemented sub-components to realize *gumonji/Q*. As shown in this figure, *gumonji/Q* consists of *gumonji*'s primary components, *gumonji* Zone Server and *gumonji* Client, and *Q* interpreter. Zone Server has an environment simulator (just written as "simulator" in the figure) as its sub-component which simulates the natural environment and maintains all objects, such as human-controlled avatars, plants, animals, on the environment. A user can access the Zone Server from a *gumonji* Client on each computer, and control an avatar on the environment. To put it concretely, user's input data is converted to a sequence of operators to control an avatar based on *gumonji*'s User API. *Q* interpreter, which transforms a *gumonji* character into an autonomous agent, is connected to the Zone Server. In order to achieve a connection between *gumonji*'s Zone Server and *Q* interpreter, we implemented sub-components, "Agent Connector" and "Q Connector", within the Zone Server and *Q* interpreter, respectively. These components communicate via TCP/IP so that *gumonji*'s Zone Server and *Q*

interpreter can be connected each other. A Q -controlled `gumonji`'s character can act as an autonomous agent. Note that a crucial point is that the environment simulator in Zone Sever can deal with human-controlled avatars and Q -controlled agents in a unified way.

As shown in Figure 2 when Q scenario, which describes an interaction pattern, is input, the scenario is converted to an execution request for Cue and Action by the scenario interpreter. The request is sent to Agent Connector on the Zone Server from Q Connector via TCP/IP. Of course, `gumonji` cannot interpret the request from the Q interpreter. Thus, we implemented a sub-component, Scenario Translator, which translates an execution request for Cue and Action to an operator sequence. As shown in Figure 2 the input from User API and Scenario Translator is identical. Therefore, an environment simulator can deal with human-controlled avatars and Q -controlled agents in a unified way. The Scenario Translator first translates the execution result to the format which is available to the Q interpreter. Then, the result is sent to Q -interpreter via Agent Connector. As stated above, a participatory simulation among human-controlled avatars and Q -controlled (autonomous) agents can be achieved on `gumonji/Q`.

3.2 Functions of `gumonji/Q`

`gumonji` enables us to simulate the natural environment, and because of its game-quality graphics, it can provide enough visual information for users. However, on `gumonji`, users cannot define diverse interactive behaviors for any `gumonji` objects. Therefore, we need additional functions to conduct participatory simulations. `gumonji/Q` makes it possible to control any objects through Q scenario so that we can construct context-sensitive agents for the simulations.

`gumonji/Q` achieves the participatory simulation environment because of its features as a network game. Users can join in simulations from their own computer environment via the Internet and they can just enjoy playing games. Therefore, it is useful for us to assemble a large number of users for simulations; this in turn allows us to obtain diverse behavior models. As we mentioned in section 2.3, on `gumonji`, we can connect multiple simulation environments via P2P connection. Thus, it is possible to realize a large-scale online simulation environment using `gumonji/Q`.

4 An Example of Participatory Modeling Using `gumonji/Q`

In this section, we conduct a participatory modeling using `gumonji/Q` to demonstrate its practicality for participatory modeling.

4.1 Participatory Modeling Process

We conduct a modeling according to a participatory modeling process proposed in 5. The process consists mainly of three steps; gaming simulation, interaction design, and multiagent simulation. The detail of each step is as follows:

1. Gaming Simulation

This step is for extracting information for constructing behavior models at the next step. In this step, we conduct a gaming simulation which reproduce a social system of a certain application domain. The process of the gaming is maintained as log data which is used next in the interaction design step.

2. Interaction Design

This step is for constructing an initial behavior model based on the log data in the first step. A behavior model is represented as a set of interaction pattern for the current state. In this step, we interview the users in order to extract the information about their decision making. Using the result of interview result and log data, we construct an initial behavior model.

3. Multiagent Simulation

This step is for verifying and improving the constructed model. We implement the model based on Q . Through multiagent simulations with implemented models, we try to verify the performance of the models and compare with the result of the gaming. If unreasonable behavior or interaction are observed during the simulations, we try to modify the models, then conduct the simulation again with the modified models.

4.2 An Overview of the Modeling Experiment

We conducted a participatory modeling using gumonji/ Q for dealing with the farmland accumulation in an agricultural colony.

The Setting of Simulation Environment. We used a virtual agricultural colony in a farmland represented by 4x4 mesh (Figure 3). Figure 3(a) shows an initial condition of a simulation. Each block is a unit of the farmland and is assigned to a farmer. For example, a block on the top left is assigned to a farmer A. For this experiment, we defined three types of farmers as shown below:

- Authorized farmer: This type of farmer actively accumulates farmland as much as possible for the efficiency of his/her work. In Figure 3(a), A and B represent this type of farmers.
- Generic farmer: This type of farmer tries to keep the current state. In the figure, C represents this type.
- Subsistence farmer: This type of farmer tends to abandon or lend his/her farmlands when the agricultural work becomes hard. In the figure, D and E represent this type.

4.3 The Process of Modeling Experiment

We explain the modeling process on gumonji/ Q step by step, and show the result. In this experiment, five users participated in the participatory modeling. Each of users performed according to the assigned type of farmers.

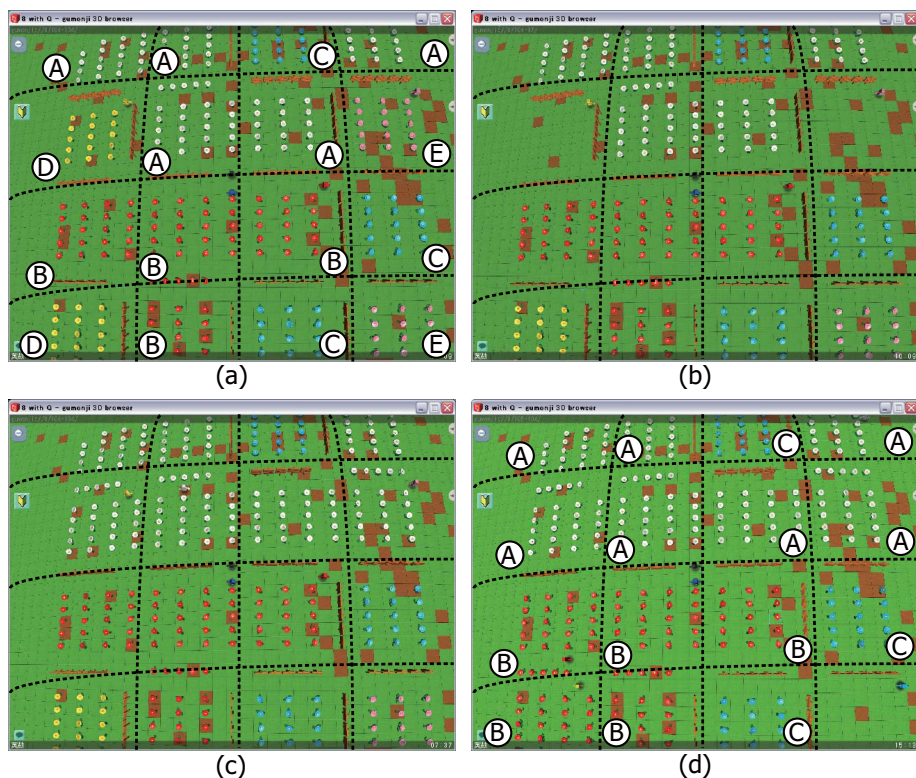


Fig. 3. Transition of the Ownership

Gaming. We first try to observe users' behaviors through the gaming in the virtual farmland. The time period of the gaming was set to 10 years. During the gaming, participants made a decision about how to manage their own farmlands, negotiate with others on the deal in farmlands, or just cultivate. During the negotiation, a borrower presents his/her desired tenancy rate, then an owner responds whether the rate is acceptable or not. If the negotiation is successfully completed, the borrower pays the fixed tenancy rate every year. The income is calculated based on the amount of harvest and rental fee. The expense is the summation of working cost, machine maintenance cost, and tenancy cost. We assume that authorized farmers own farm machines, so that their working cost are reduced by using machines, but they must pay machine maintenance cost. Each user select behavior to maximize utility.

The transition of the owner of farmlands is shown in Figure 3. During the gaming, we could observe several interaction between users, such as mentioned below. A subsistence farmer D had continued to cultivate until the middle of the period, but after that, he stopped cultivating because it became difficult for him to maintain his farmland. Thus, an authorized farmer A offered D to barrow the abandoned farmland. Although A also offered C to barrow his farmland, C refused

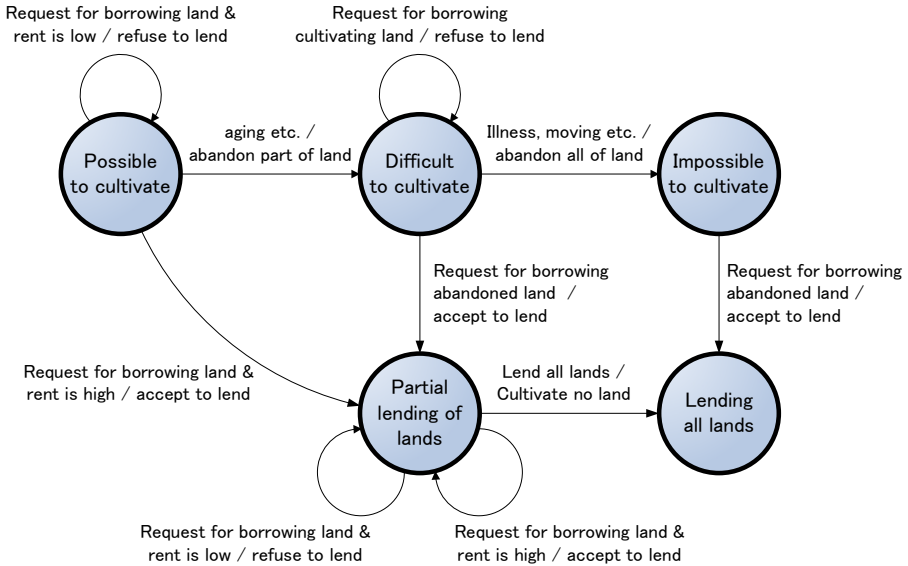


Fig. 4. A Behavior Model of a Generic Farmer

the offer. As a result, the owner of farmlands became as shown in Figure 3(d). In the end of the gaming, a farmer E abandoned his/her farmland (bottom right block in the figure).

Interaction Design. We extracted the interaction pattern from the observation obtained in the gaming. For the extraction of interaction pattern, we interviewed the users and asked them how they made a decision in each state. With the obtained interaction patterns, we constructed behavior models of each user. As we mentioned above, a behavior model is constructed as a set of interaction patterns.

Figure 4 shows an obtained model for a generic farmer. In this model, a farmer abandons some parts of his/her farmlands when it becomes difficult for him/her to continue to cultivate, and he/she abandons all his/her farmlands when it is impossible to maintain farmlands due to any reasons, such as aging, and so on. Additionally, this model shows that a farmer accepts an offer to borrow the abandoned farmland.

Multiagent Simulation. The obtained behavior models were assigned to the agents and multiagent simulations were executed. We observed simulations, then compared the result of the gaming and simulations. Based on the comparison, we modified behavior models when necessary. For example, in the simulations, a generic farmer always refused offers to borrow his farmland even when he was in the, “difficult to cultivate” state. This was because there was no edge to accept an offer to borrow “not abandoned” farmland with any tenancy rate. Figure 5. In this model, there are two edges to accept an offer with the expensive rate and refuse an offer with the cheap rate.

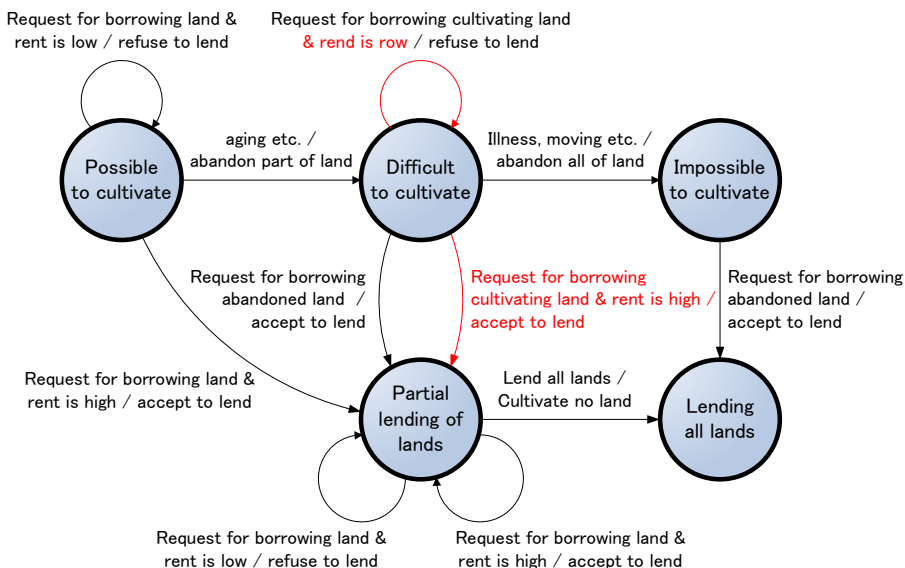


Fig. 5. An Improved Behavior Model of a Generic Farmer

We obtained models from the observation of users' behaviors by participatory simulation on *gumonji/Q*. We consider that this result verifies the practicality of *gumonji/Q* for participatory modeling.

5 Conclusion

In this paper, we developed a novel networked participatory simulation platform, *gumonji/Q*, by integrating scenario description language *Q* and network game *gumonji*. In *gumonji/Q*, an user can design interaction pattern between entities using *Q*. An user also perform simulation on virtual space provided by *gumonji* where he/she can interact with the environment and make a decision based on game-quality graphics. In order to connect *Q* and *gumonji*, we implemented communication sub-components for achieving TCP/IP communication between them, and a scenario translator to convert a request from *Q* to a sequence of operators. Then, we showed an example of how to obtain human behavior models through a simulation on *gumonji/Q*.

We developed a PMAS platform in which many humans can participate and interact with agents and the surrounding environment practically. By using *gumonji/Q* as a simulation platform, we can expect that users are more enticed to participate in simulations on *gumonji/Q* since simulations on *gumonji/Q* seems more enjoyable than a directive simulations. Additionally, because of *gumonji/Q*'s features as a networked simulator, they can access to the simulation from their own computer environment via Internet.

Our future works include developing a participatory modeling methodology which is suitable for distributed participatory simulation environment.

References

1. Axelrod, R.: Advancing the art of simulation in the social sciences. *Complex* 3, 16–22 (1997)
2. Drogoul, A., Ferber, J.: Multi-agent simulation as a tool for modeling societies: Application to social differentiation in ant colonies. In: Castelfranchi, C., Werner, E. (eds.) *MAAMAW 1992*. LNCS, vol. 830, pp. 3–23. Springer, Heidelberg (1994)
3. Murakami, Y., Sugimoto, Y., Ishida, T.: Modeling human behavior for virtual training systems. In: *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pp. 127–132 (2005)
4. Gilbert, N., Maltby, S., Asakawa, T.: Participatory simulations for developing scenarios in environmental resource management. In: *Third workshop on agent-based simulation*, pp. 67–72 (2002)
5. Torii, D., Ishida, T., Bousquet, F.: Modeling agents and interactions in agricultural economics. In: *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pp. 81–88 (2006)
6. Colella, V., Borovoy, R., Resnick, M.: Participatory simulations: using computational objects to learn about dynamic systems. In: *CHI 1998: CHI 1998 conference summary on Human factors in computing systems*, pp. 9–10 (1998)
7. Ishida, T.: Q: A scenario description language for interactive agents. *Computer* 35, 42–47 (2002)
8. Nakanishi, H., Ishida, T.: Freewalk/q: social interaction platform in virtual space. In: *VRST 2004: Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 97–104 (2004)
9. Bousquet, F., Bakam, I., Proton, H., Page, C.L.: Cormas: Common-pool resources and multi-agent systems. In: *IEA/AIE 1998: Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pp. 826–837 (1998)
10. Murakami, Y., Ishida, T., Kawasoe, T., Hishiyama, R.: Scenario description for multi-agent simulation. In: *AAMAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 369–376 (2003)
11. Nakanishi, H., Nakazawa, S., Ishida, T., Takanashi, K., Isbister, K.: Can software agents influence human relations?: balance theory in agent-mediated communities. In: *AAMAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 717–724 (2003)
12. Ishida, T.: Society-Centered Design for Socially Embedded Multiagent Systems. In: Klusch, M., Ossowski, S., Kashyap, V., Unland, R. (eds.) *CIA 2004*. LNCS, vol. 3191, pp. 16–29. Springer, Heidelberg (2004)

A Multiagent-System Framework for Hierarchical Control and Monitoring of Complex Process Control Systems

Vu Van Tan*, Dae-Seung Yoo, and Myeong-Jae Yi

School of Computer Engineering and Information Technology
University of Ulsan, San-29, Moogu-2 Dong, Namgu, Ulsan, 680-749, Korea
{vvtan,ooseyds,ymj}@mail.ulsan.ac.kr

Abstract. This paper proposes a framework for the implementation of multiagent system for hierarchical control of complex process control systems based on OPC¹ technology that is widely applied to the automation control systems. This framework is proposed with utilization of OPC technology in both continuous-event part and discrete-event part by incorporating with XML for the negotiation and cooperation in the environments of multiagent system. The framework design criteria are also described. The comparison of the proposed framework with existing frameworks is made to demonstrate that the proposal is reliable and feasible in order to apply to agent-based process control applications.

Keywords: Hierarchical control, multiagent, OPC, process control, XML.

1 Introduction

Automation and information systems designed for industrial plant floor are more complex and large, including a lot of different components and different platforms such as control instrumentations, control softwares, etc. However, the agent technology could be help for them as mentioned in several researches [2,9,4]. A complex process control system should place on different hardware and software and accomplishes its tasks. The problem deals with the appropriate synthesis of flexible mechanism for accessing and updating the process data, events, operator and external expert decisions, and negotiations. Control tasks for complex process control systems are still challenging because of the complexity of related decision tasks and information systems. It seems that they are hard to integrate more advanced control strategies into real-world situations because of the widely used software systems in enterprises and integration of process control systems.

Although a number of existing approaches for the agent-based complex process control systems have been successfully proposed and developed in recent years

* Corresponding author.

¹ Openness, Productivity, and Collaboration; formerly “OLE for Process Control”.

[2,9,4,17,16,12,8,11,3,6,5,10,1], it seems that they are still far away from a ultimate solution. These approaches also have limited in attention to the monitoring operations [14]. Research on applications of multiagent system in process control and monitoring has been less extensive than in discrete manufacturing. The reason is that both the suitability and usefulness of multiagent system in process control are maybe not evident than in discrete manufacturing. Nowadays, the OPC Foundation has defined a new specification as the next generation for process control and monitoring running on various platforms, i.e., a series of the OPC Unified Architecture (UA) specifications [15]. It provides a paradigm for the design and implementation of control softwares.

The aim in this paper is to propose a framework for the implementation of multiagent system for hierarchical control and monitoring of complex process control systems based on OPC UA technology. The framework-based system allows to access and update the process data, events, operator and external expert decisions. This framework is designed with self-organizing database to provide a flexible method when applying to various applications.

This paper is organized as follows. The next section introduces the framework design criteria. Section 3 proposes an OPC-based framework architecture of multiagent system for hierarchical control and monitoring. To provide the ability of storing data, events, and parameters, the design of database structure is represented in Section 4. The comparison of the proposed framework with existing frameworks is provided to demonstrate that the proposed framework is reliable and feasible to apply to various real applications as presented in Section 5. In short, Section 6 will mark some conclusions and future work.

2 Framework Design Criteria

An object-oriented framework was defined as a prefabricated extensible set of classes or components with predefined collaboration between them and extension interfaces [5]. A software framework for multiagent system has to meet the requirements that rise from the agent point of view. A two-step requirement analysis to construct the framework criteria should be performed including *domain analysis* and *domain design* [10]. The framework design criteria are therefore specified as follows:

1. *Generic requirements.* The aim of the proposed framework is the process control system domain. A universal architecture is therefore necessary with treating different organizational types of process control systems.
2. *Methodology.* The concept of the proposed framework introduces a level of abstraction that provides an easier conceptualization of the problem domain. This enables the implementation of more complex control strategies.
3. *Flexibility.* The flexibility of the proposed framework is incorporated in the context of hierarchical control and monitoring of process control systems.
4. *Scalability.* It is relatively easy to add new agents to a multiagent system and to change parameters for controlling the states of control devices.

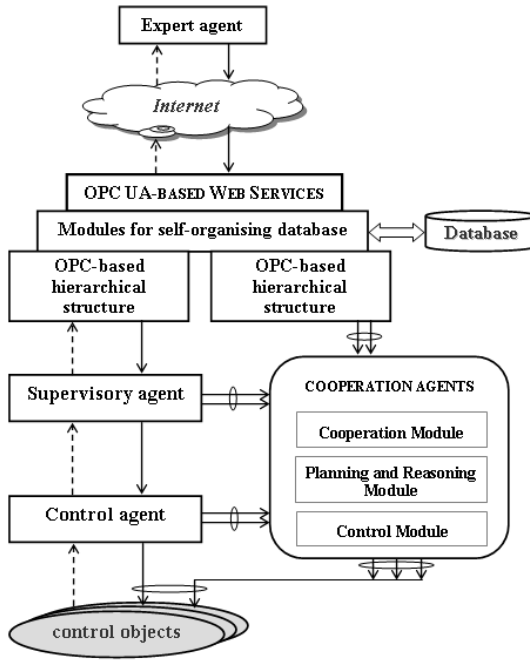


Fig. 1. The proposed architecture of multiagent system and its agent levels

5. *Reusability.* This framework is ensured by developing a generic structure, level of agents, and database structure for storing the control parameters, process data, events, operator and external expert decisions.
6. *Openness.* The proposed framework is developed according to OPC technology to makes it flexible and open for agent-based complex process control systems to develop various agent applications. It should comply with the specification of FIPA standards² for agent technology as close as possible.
7. *Adaptivity.* Agents may have adaptive behaviour, i.e. the ability of learning from experience about the environments to improve the choice of future actions. The control parameters are stored in additional tables and automatically restored by the proposed framework when client starts.

3 Design of Multiagent-Based Process Control System

3.1 System Architecture

The type of an agent depends on the way of the state changes and on the knowledge of the agent. The agents are specified on a design level in an iterative process in addition of knowledge. The structure of agents of the hierarchical control system can be shown in Fig. 1. It consists of the three kinds of the agents such as *control agent*, *supervisory agent*, and *expert agent* [2].

² <http://www.fipa.org/>

The agents in the multiagent systems not only communicate with the users and objects, but also communicate and incorporate with others. To solve these problems, the direct control interaction can be used with cooperation of control system functions. In addition, the distributed control systems (DCS) is normally divided into two types of behaviors such as time-driven and event-driven, respectively. Each agent might initiate a state transition, but not every agent might actually enforce this agent without cooperation with other agents.

The most important issue for multiagent systems is to cooperate the agents and to communicate an agent with others. In the cooperation model provided by the different levels of abstraction, the mappings of real objects of the controlled objects can be presented in *Address Space* based on OPC technique as a hierarchical structure. Moreover, the supervisory agent is equipped with models, methods, and algorithms to additional indirect controls.

The architecture of a multiagent system based on OPC technology with methods, knowledge of expert agent, and needful components is proposed as shown in Fig. 1. It indicates how the hierarchical control levels are presented and provides detail information on guarantees of two types of the behaviors in DCS.

3.2 Control and Monitoring Operations

The multiagent-based process automation system, i.e., process control and monitoring system, consists of two types of agents: 1) process agents and 2) monitoring agents. The process agents perform supervisory control operations either in sequential or iterative fashion. This relates to process state change or batch control operations and the later to tuning of continuous control. The task of the agent is first to plan a shared sequence of control actions and then to execute this sequence. It is to calculate optimal values for the supervisory control variables. Therefore, the decision-making processes of the agents are different depending on the role of the iterative refinement of the supervisory control variables. The results from the decision-making tasks are new values of control variables.

The monitoring agents perform monitoring operations in a distributed manner. Their operation has foundation by combining information from field-device measurements, operational state classification, simulations, condition monitoring, and process models. The operations of the monitoring agents are based on distributed search, processing, and monitoring of information. The search of information is decomposed using the understanding of physical structure of the monitored process, its present state, various diagnostics reports, and ability of different information providers.

3.3 Agent Model for the Control and Monitoring Operations

The agent model of an automation agent³ specifies the internal modules of an agent and its operations. The automation agents also need to be conformed to the agent model of some FIPA-compliant generic agent platform. In general, an agent mainly consists of a set of behavior and action to define the agent reaction

³ It is used to either indicate a process agent or a monitoring agent.

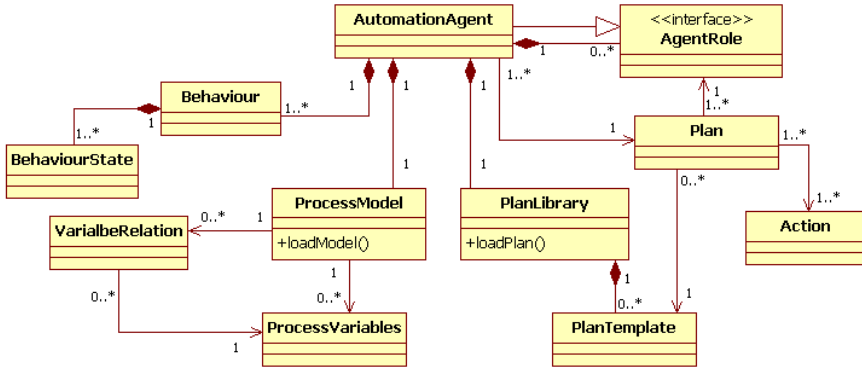


Fig. 2. The UML class diagram for the process automation agent

to different circumstances like incoming messages or events. Agent behaviors, which form the basis for reactivity and pro-activity [10], are used to implement agent interactions. A process automation agent will use its actions to fulfill its goals. The automation agent is composed of modules that can be classified into operational and modeling modules and runtime data structures. The agent model used for the proposed framework is illustrated in Fig. 2.

The process model describes the knowledge of a process automation agent on the controlled process. A process automation agent can have knowledge about the existence of process variables, measured values, and relation between variables. A control variable may be controlled by one agent whereas the measurements can be shared among several agents. The plan library contains plans that a process automation agent can use during planning in order for creating runtime plans.

4 Database Structure

Because agents require accessing to database to be able to participate in an architecture where information is exchanged, the connection between the proposed framework and relational database standards must be guaranteed. The information generated by the OPC server-client model is stored in the additional tables created by the specific database. The additional tables and their relationships are designed as shown in Fig. 3 to store the data, events and other information. Other tables can be added to this database structure. It ensures that various real applications can be satisfied adequately.

When the administrator is to terminate a choice of the interesting items, i.e., objects, for reading and writing and then control algorithm is also defined for operating on items that are established in the OPC server for reading or calculating values. Items selected by the expert agent constitute the expert configuration and should be saved in the database. The saved configuration will be automatically restored when the browser-based client starts. While the expert agent processes the defined configuration, it can set to the *read/write* mode in

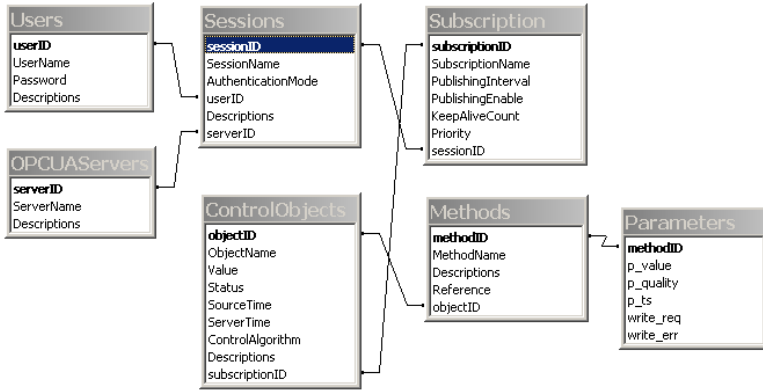


Fig. 3. The tables and relationships among the additional tables in the relational database for the proposed framework

the application. The appropriate mechanisms in order to the execution of the proposed system with control algorithms are included the following steps:

1. Downloading the current values from the additional tables designed in the database and then storing these values to the control variables.
2. Executing the control algorithms to calculate the new values that will be stored in the database, i.e., data or events and parameters are updated to the additional tables automatically.
3. Writing the changed values from the additional tables into the control objects represented at the *Address Space* configured by the administrators.
4. Sending the calculated values created by the control algorithms to the OPC server-client model. These values depend on the specific control algorithms.

5 Comparison with Existing Frameworks

To compare the proposed framework with other approaches, it is difficult to conduct a fair comparison of different proposals and architectures because of their conceptual nature and wide range of production environments. A qualitative comparison of the proposed architecture and existing frameworks could be made. The structural characteristics of the proposed system are therefore used to compare with others. The proposed framework is designed for various process control system while some existing approaches focus only on the development of specific applications [24,11,6]. It has a number of advantages as follows:

1. The control address space can collect current data and events that are truly related to each others by using new features of the OPC UA specification. The concept of the proposed framework introduces a new level of the design and implementation in order to provide an easier conceptualization of the control agent domain. It enables the implementation of more complex control strategies where the control is distributed. It is also suitable for developing control strategies where large amounts of distributed data are collected.

2. The proposed framework is mainly for process planning, scheduling and control with integrating the hierarchical and decentralized controls which make the proposed system flexible. But existing approaches are entire manufacturing systems that include product development, product planning, process planning generation, and scheduling and control, e.g., PROSA [1].
3. Data or events are automatically stored to the relational database. The database structure is open for the application developers.
4. Unlike several existing approaches that focus on the domain of manufacturing systems such as mentioned in [7,8,6,10,11], the domain of this framework is that the hierarchical control and monitoring of complex process control systems is covered including the guarantees of updating the process data, events, operator and external expert decisions, and negotiation.
5. The structure of the proposed framework is designed and developed in a systematic way. This framework has focused not only on the control functions, but also on other functions like monitoring operations. The modules implemented within the framework are used for the purpose of developing the agent-based process control systems and new roles for agents.

6 Concluding Remarks and Future Work

This paper has introduced a framework for the implementation of multiagent system for hierarchical control and monitoring of the complex process control domain. Based on the requirements of the process control domain, the framework design criteria were proposed. A generic architecture was suggested for the implementation of the multiagent-based process control system that allows for a flexible integration of different control strategies. Details about architecture, agent model, and database structure are presented. As a result, the proposed framework fulfills the seven framework design criteria. It was proposed for providing a design foundation and a common framework to realize its potential to the application developers when implementing various agent applications.

In future research, trying to use the proposed framework for the implementation of an agent-based process control approach for flexible process control systems is the major task.

Acknowledgments. The authors would like to thank Korean Ministry of Knowledge Economy, Ulsan Metropolitan City, University of Ulsan, and the Network-based Automation Research Center (NARC) which partly supported this research. The authors also thank the anonymous reviewers for their carefully reading and commenting this paper.

References

1. Brussel, H.V., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. *Computers in Industry* 37, 255–274 (1998)

2. Choinski, D., Nocon, W., Metzger, M.: Multi-Agent System for Hierarchical Control with Self-organising Database. In: Nguyen, N.T., Grzech, A., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2007. LNCS, vol. 4496, pp. 655–664. Springer, Heidelberg (2007)
3. Damba, A., Watanabe, S.: Hierarchical Control in a Multiagent System. In: Proceedings of the 2nd International Conference on Innovative Computing, Information and Control, ICICIC 2007, p. 111 (2007)
4. Davidsson, P., Wernstedt, F.: Software Agents for Bioprocess Monitoring and Control. *Journal of Chemical Technology and Biotechnology* 77(7), 761–766 (2002)
5. Fayad, M.E., Schmidt, D.C., Johnson, R.E. (eds.): *Building Application Frameworks: Object-Oriented Foundation of Framework Design*. Wiley, New York (1999)
6. Guo, Y., Cheng, J., Gong, D., Zhang, J.: A Novel Multi-agent Based Complex Process Control System and Its Application. In: van Leeuwen, J. (ed.) WG 1988. LNCS, vol. 344, pp. 319–330. Springer, Heidelberg (1989)
7. Heragu, S.S., Graves, R.J., Kim, B.I., Onge, A.S.: Intelligent Agent Based Framework for Manufacturing Systems Control. *IEEE Transactions on Systems, Man, and Cybernetics - Part A* 32(5), 560–573 (2002)
8. Leduc, R.J., Lawford, M., Dai, P.: Hierarchical Interface-Based Supervisory Control of a Flexible Manufacturing System. *IEEE Transactions on Control Systems Technology* 14(4), 654–668 (2006)
9. McArthur, S.D.J., Davidson, E.M.: Multi-Agent Systems for Diagnostics and Condiation Monitoring Applications. In: Proceedings of the 13th International Conference on Intelligent System Application to Power Systems, pp. 201–206 (2005)
10. Monch, L., Stehli, M.: ManufAg: A Multi-agent-System Framework for Production Control of Complex Manufacturing Systems. *Information Systems and e-Business Management* 4(2), 159–185 (2006)
11. Najid, N.M., Kouiss, K., Derriche, O.: Agent based Approach for a Real-time Shop Floor Control. In: Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp. 6–9 (2002)
12. Nocon, W., Choinski, D., Metzger, M.: Web-based Control and Monitoring of the Experimental Pilot Plant Installations. In: Proceedings of the IFAC Workshop on Programmable Devices and Systems, pp. 94–99 (2004)
13. Seilonen, I., Pirttioja, T., Pakonen, A., Appelqvist, P., Halme, A., Koskinen, K.: Information Access and Control Operations in Multi-agent System Based Process Automation. In: Mařík, V., William Brennan, R., Pěchouček, M. (eds.) HoloMAS 2005. LNCS (LNAI), vol. 3593, pp. 144–153. Springer, Heidelberg (2005)
14. Seilonen, I., Pirttioja, T., Pakonen, A., Appelqvist, P., Halme, A., Koskinen, K.: Modelling Cooperative Control in Process Automation with Multi-agent Systems. In: Proceedings of the 2nd IEEE International Conference on Industrial Informatics, INDIN 2004, pp. 260–265 (2004)
15. The OPC Foundation. OPC Unified Architecture Specification: Parts 1-11. Version 1.xx (2006-2007), <http://opcfoundation.org/>
16. Thomas, S.J., et al.: Monitoring and Analysis of Multiple Agent Systems. In: Proceedings of the NASA/JPL Workshop on Radical Agent Concepts, <http://www.psatellite.com/papers/wrac2001.pdf>
17. Wooldridge, M.: *An Introduction to Multiagent Systems*. Wiley, Chichester (2002)

Agent Reasoning with Semantic Web in Web Blogs

Dinh Que Tran and Tuan Nha Hoang

Faculty of Information Technology
Posts and Telecommunications Institute of Technology
Km10, Nguyen Trai, Hanoi, Vietnam
tdque@yahoo.com, tuannhahoang@yahoo.com

Abstract. The Web pages contain lots of useful information but their complex layouts, unstructures and semantics are becoming obstacles for autonomous software agents in querying as well as processing. Various studies for representing web information as well as reasoning to infer useful knowledge are active topics in semantic web. This paper focuses on presenting an architecture of a multi-agent system with reasoning ability in web blog domain. The system includes classes of agents: crawler agents extract relevant data from various web blog resources and then convert it to the form of OWL, reasoning agents makes use of these resources and reasoning mechanism to infer necessary information so that user agents get and dispatch the result to mobile phone user.

Keywords: Semantic web, reasoning, information integration, ontology, multi-agent system, web blogs, web mining.

1 Introduction

A large number of pages on the World Wide Web are “personal home pages” or “web blogs” that are concerned with personal information such as biographies, jobs, interests, hobbies. Among them are Yahoo blog, blogger.com and social networks such as Myspace.com, Facebook.com, and Blogs.ebay.com. The information resources on these blogs have been exploiting for recruiting talents, comparing prices in e-commerce, hunting jobs and so on. However, their heterogeneous properties of formats, structures and semantics have become obstacles for autonomous agents in integrating these information contents.

Using various technologies of semantic web in enabling software agents to syndicate and share information is active topics in web mining. The important advantage of semantic web is the ability of combining reasoning into agents to infer new knowledge from a given set of information ([1], [5], [7], [6], [8], [11], [15], [16], [18], [19]). Studies on reasoning focus on applying reasoning techniques that have been widely investigated in AI such as rule-based reasoning, plausible and non-monotonic reasoning or probabilistic reasoning.

This paper presents an architecture of multi-agent system that is able to extract heterogeneous information on the web blogs, to convert to OWL format and then to make use of a reasoning mechanism to deduce new information in web blogs. The paper is

structured as follows. Section 2 presents a reasoning of semantic web. Section 3 is the architecture of the multi-agent system and Section 4 presents conclusions.

2 Reasoning in Semantic Web

2.1 OWL Overview

Semantic Web is considered as web of data resources that can be processed directly or indirectly by machines. OWL (Web Ontology Language) ([2], [3], [5], [13], [15], [18]) is the standard in data representation of semantic web and is used in need of processing the content of information rather than displaying information such as HTML. OWL is an evolution of the precedent languages DAML+OIL, which is based on Description Logic. An illustration of the evolution is given in Figure 1 [2]:

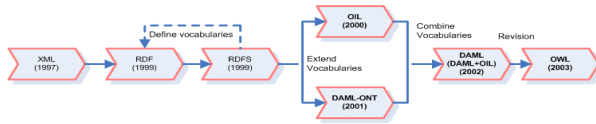


Fig. 1. Evolution of OWL

Being an extension of RDF (the Resource Description Framework) for a general model in representing the resources on the web through triples of “subject”, “predicate” and ”object”, OWL supports ([2], [3]):

- Creating information structure to be understandable and usable by agents
- Making the possibility to reuse knowledge for applications in different contexts
- Providing a tool for developing and adapting the current knowledge in changing environment
- Integrating the information resources into a larger knowledge base.

An example on Web Blogs Domain is represented in OWL (Table 1):

Table 1. OWL representation of Web Blog Domain

```

<owl:Ontology rdf:about="Person Information"/>
<owl:Class rdf:ID="Person"/>
<owl:Class rdf:ID="Company">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Association"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasFriend">
  <rdfs:range rdf:resource="#Person"/>
  <rdfs:domain rdf:resource="#Person"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="interestedIn">
  <rdfs:range rdf:resource="#Interest"/>
  <rdfs:domain rdf:resource="#Person"/>
</owl:ObjectProperty>
.....

```

2.2 Reasoning Rules in OWL

Reasoning based on rules has been familiar in AI community and is widely used in developing multi-agent systems. Reasoning both in a description logic knowledge and rule-based knowledge has gained much interest in the Semantic Web community ([11], [12], [15], [16], [19], [20]). An illustration of representation of rule in OWL is given in Table 2.

Table 2. Representation of rule

<p>Example Query: who has the interest in Information Technology? Rule: If a person? P has hasInterests is ?M, and ?M is one instance of Information Technology. Instances of Information Technology include Computer Science, Software Engineering, Agent Technology... [likeInformatics: (?P interestedIn ?M), (?M hasType Informatics)-> (?P interestedIn Informatics)]</p>
--

3 Information Query System on Web Blogs

3.1 Scenario

Consider the following scenario:

A client would like to use mobile phone to seek people who have the same hobbies or some characteristics on her demand.

The above problem may be reduced to the following sub-problems: (1) Extracting personal information from blog pages such as Yahoo blog, blogger.com. And then converting it into a format OWL; (2) Reading the formatted data and performing inference based on ontology and constructed rules; (3) Integrating the resulted information and sending to users. Solvers for these problems are software agents whose architecture and functions are described in the next section.

3.2 System Architecture

The multi-agent system is built and implemented by using Jade platform [4] and its architecture is described in Fig. 2. The system collects information from web resources and stores them in OWL format. The information will be combined with the inference rules and ontology in order to produce new knowledge for querying later.

Crawler engine: Collecting the blog pages containing personal information, and to download these HTML documents to local host. Since most blogs connect to each other by linking “friend”, we can crawl a large number of blogs from a set of seed URLs. Our crawler engine is composed of agents, which is extended from multi-thread model [20], to allow autonomous agents to locate on various machines. The `CoordinatingAgent` maintains a common list of unvisited URLs that is initialized with seed URLs provided by a user or another program. Each `CrawlingAgent` loop involves picking the next set of URLs to crawl from the common queue, fetching the page corresponding to the URL through HTTP, parsing the retrieved page to extract the URLs and specific information, and finally adding the unvisited URLs to the `CoordinatingAgent` (Figure 3).

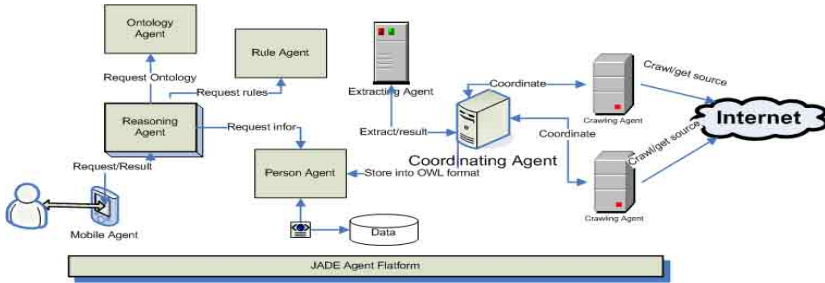


Fig. 2. Multi-agent system architecture

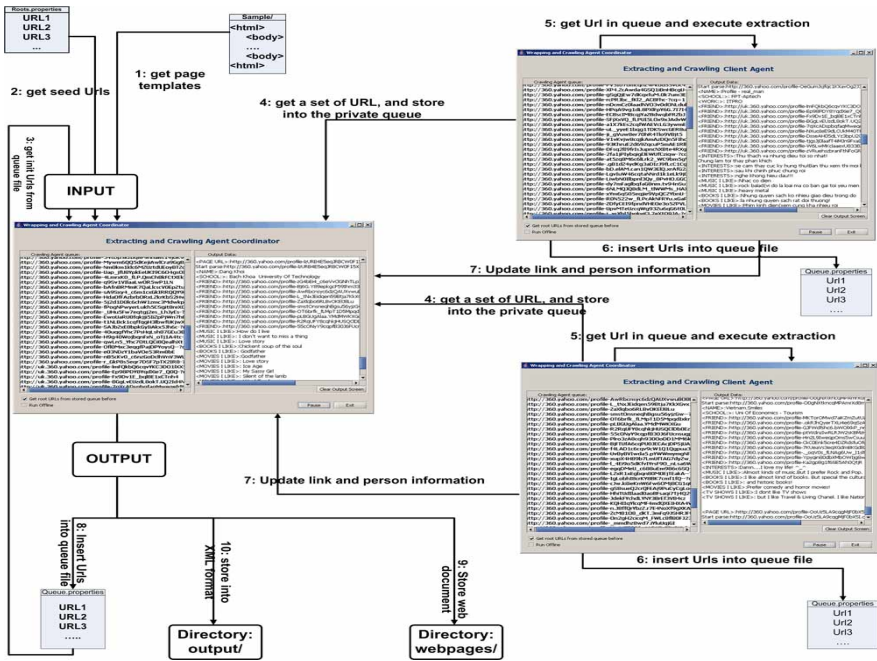
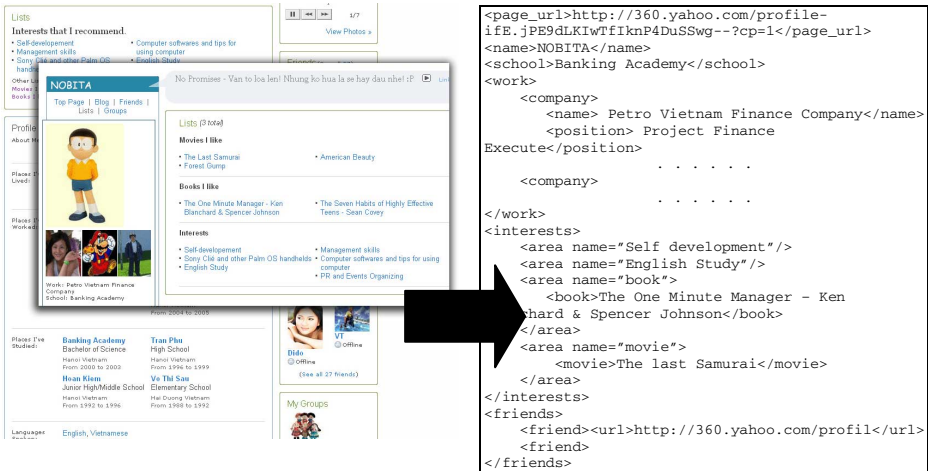


Fig. 3. Crawling Agent architecture

Extracting Engine: Information is extracted from blog pages by using grammatical inference technique ([14], [17]), and then the collected data is normalized in XML format in predefined template and converted to OWL format (Fig. 4).

Reasoning engine: Including **PersonAgent**, **DefaultOntologyAgent**, **DefaultRuleAgent**, **ReasoningAgent** (RA) and other **OntologyAgents**, **RuleAgents**. It takes over the responsibility to integrate rules, ontology and the collected information to generate new information.

- **PersonAgent** : dispatching personal data in OWL to **ReasoningAgent** as well as supplying a query engine to query OWL data from its database. **PersonAgent** performs actions:



The image shows a screenshot of a web profile page for a user named 'NOBITA'. The page includes sections for 'Interests that I recommend', 'Movies I like', 'Books I like', and 'Interests'. An arrow points from the 'Interests' section to an XML representation of the profile data.

```

<page_url>http://360.yahoo.com/profile-
ife.jPE9dLKIwTfIkP4DuSSwg--?cp=1</page_url>
<name>NOBITA</name>
<school>Banking Academy</school>
<work>
  <company>
    <name> Petro Vietnam Finance Company</name>
    <position> Project Finance
Execute</position>
  <company>
    . . . . .
</work>
<interests>
  <area name="Self development"/>
  <area name="English Study"/>
  <area name="book">
    <book>The One Minute Manager - Ken
ard & Spencer Johnson</book>
  </area>
  <area name="movie">
    <movie>The last Samurai</movie>
  </area>
</interests>
<friends>
  <friend><url>http://360.yahoo.com/profil</url>
<friend>
</friends>

```

Fig. 4. Extracted result in XML format

Begin: waiting request for OWL data from RA

Update: updating new set of data, this is received from RA or Extracting/Coordinating Agent, to its database

Query: when receiving “query” message from RA, it starts executing query and then send the result to RA

- DefaultOntologyAgent receive request from RA, if the needed ontology is not available, it will contact with OntologyYellowpageAgent to find which OntologyAgent supports ontology in required domain. When ontologies are received from different resources, they must be integrated by OntologyAgent before being sent to RA. DefaultOntologyAgent performs behaviors:

Begin: waiting state for receiving a request from RA and other Ontology Agent.

Find: when receiving “ontology” message from RA, it checks if its domain is corresponding to the request message. If not, it will send “search” message to Yellow page OntologyAgent to get address of all OntologyAgents supporting ontology in this domain.

Exchange: sending an “exchange” message to other Ontology Agent to get necessary data.

End: The ending is successful if there is at least one agent that supports ontology in the request domain. Then the result will be sent back to RA.

- DefaultRuleAgent supplies inference rules for RA, if RA requests rules that are not available in its database, it will send request to RuleYellowpageAgent for seeking suitable rules. The received rules must be pre-processed, integrated and then sent to RA. DefaultRuleAgent behaviors are similar to DefaultOntologyAgent.

- Mobileagent:** using Jade-leap [4], this agent will locate on users' mobile and connect with ReasoningAgent through Jade platform (Figure 5). It supports clients to query and display the result. When starting query by Search, the request is encoded into a query statement, then sending to ReasoningAgent. RA will process this query and then execute reasoning based on its knowledge and return the expected result. The terminal device will display the information from ReasoningAgent.

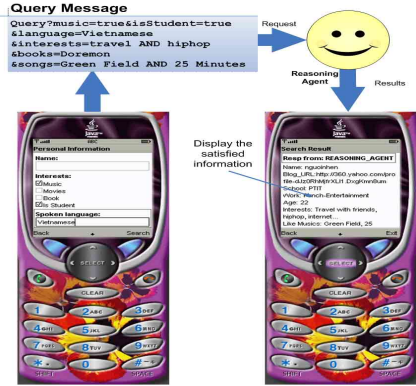


Fig. 5. Query and result

- ReasoningAgent** is the important component of our system. It is developed based on Jena ([6], [7]) that supports effective storage for a number of databases (e.g., MySQL, Oracle) and querying. ReasoningAgent supports behaviors (Figure 6):

Begin: RA is in waiting state to receive request from MobileAgent

Reasoning initiation: It checks if the domain of request message is available in its default data: Ontology and Rule. If its ontology and rules are not related to requesting domain, RA will send a request messages to DefaultOntologyAgent and DefaultRuleAgent to get the needed ones.

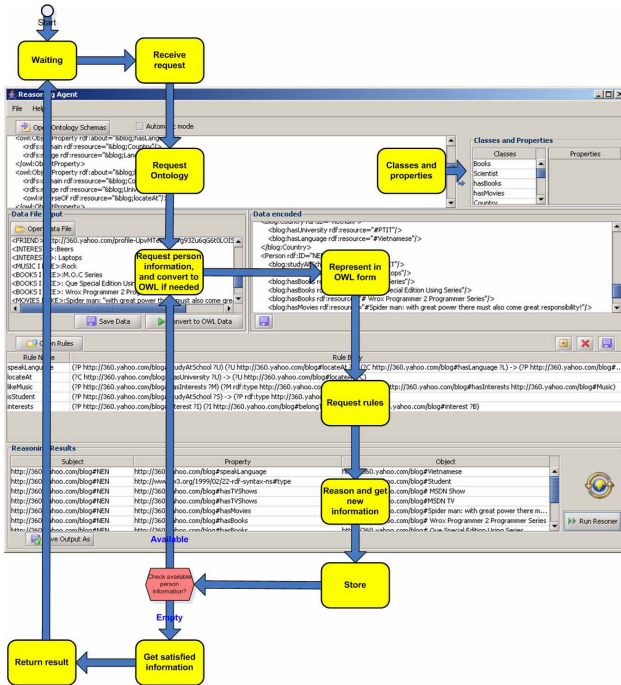


Fig. 6. Activity of Reasoning Agent

Reasoning: Receiving suitable ontology and rules, it performs the process of reasoning. It will send “person” message to `PersonAgent` to get successively sets of person information in OWL format. RA makes inference on the received information and then sends an updated message to `PersonAgent`. Lastly, sending the query result to `MobileAgent`, and then coming to waiting state.

4 Conclusions and Further Research

Our paper presented architecture of a multi-agent system which comprises a sub-architecture for crawling and extraction and sub-architecture of reasoning. The personal information is firstly extracted from web blogs and the converted into OWL format and stored in database. `ReasoningAgent` combines the ontology and rules with personal information in the OWL format to infer new knowledge in which ontology and rules are supplied from various agents. Our research is among efforts to bridge between the current web and semantic web.

With this architecture, crawling process with agents can take advantage of internet bandwidth; ontology and rules can be managed more flexible and easier for constructing. However, ontology and inference rules presented in this system is not enough large to meet the diversity of person information. We are currently studying: (i) Application of data mining techniques to automatically construct rules and ontology from OWL database; (ii) Making use of reasoning techniques such as probabilistic reasoning, non-monotonic reasoning to this web logs domain. These research results will be presented in our further work.

References

1. Antoniou, G., et al.: Reasoning methods for personalization on the semantic web. *Annals of Mathematics, Computing & Teleinformatics* (2004)
2. Horrocks, I.: Hybrid Logics and Ontology Languages. Talk at HyLo (2006)
3. Horrocks, I.: DAML+OIL: a Description Logic for the Semantic Web (2002)
4. Jade - Java Agent DEvelopment Framework, <http://jade.tilab.com/>
5. Hendler, J.: Agent and Semantic web. *IEEE Intelligent Systems* 16(2) (2001)
6. Jena project homepage, <http://jena.sourceforge.net/>
7. Jeremy, J., et al.: Jena: Implementing the Semantic Web Recommendations. Technical Report HPL (2004)
8. Kuno, K., Wilkinson, C., Reynolds, D.: Efficient RDF storage and retrieval in Jena2. In: *Proceedings of VLDB Workshop on Semantic Web and Databases* (2003)
9. Mostafa, M., et al.: The Ontology Web Language (OWL) for a Multi-Agent Understanding System (2005), <http://ieeexplore.ieee.org/iel5/9771/30814/01427149.pdf>
10. Resource Description Framework (RDF) / W3C Semantic Web Activity, <http://www.w3.org/RDF/>
11. Heymans, S., Van Nieuwenborgh, D., Vermeir, D.: Nonmonotonic Ontological and Rule-Based Reasoning with Extended Conceptual Logic Programs. In: Gómez-Pérez, A., Euzenat, J. (eds.) *ESWC 2005*. LNCS, vol. 3532, pp. 392–407. Springer, Heidelberg (2005)
12. Cayzer, S.: Semantic Blogging: Spreading the Semantic Web Meme (2004)

13. The Protégé Ontology Editor and Knowledge Base Acquisition System, <http://protege.stanford.edu/plugins/owl/>
14. Hong, T.W.: Grammatical Inference for Information Extraction and Visualisation on the Web. Ph.D. Thesis, Imperial College London (2002)
15. Wang, X.: Ontology Based Context Modeling and Reasoning using OWL. In: Proc. Of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW 2004) (2004)
16. Terziyan, V.: Semantic Web - new Possibilities for Intelligent Web Applications. Course Introduction, Department of Mathematical Information Technology, University of Jyväskylä. ITIN, France (2006)
17. Crescenzi, V., Mecca, G., Merialdo, P.: Roadrunner: Towards automatic data extraction from large web sites. Technical Report, Universit di Roma Tre (2001)
18. Haarslev, V., Moller, R.: Racer - An owl reasoning agent for the semantic web. In: Proc. of IEEE/WIC International Conference on Web Intelligence, Halifax Canada (2003)
19. Ding, Z., Peng, Y.: A Probabilistic Extension to Ontology Language OWL. In: 37 th Hawaii International Conference on System Sciences, Hawaii (2004)
20. Pant, G., Srinivasan, P., Menczer, F.: Crawling the Web. In: Levene, M., Poulouvasilis, A. (eds.) Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Springer, Heidelberg (2004)

Design of a Multiagent System over Mobile Devices for the Planning of Touristic Travels

Miguel Valdés and Claudio Cubillos

Pontificia Universidad Católica de Valparaíso, Escuela de Ingeniería Informática,
Av. Brasil 2241, Valparaíso, Chile
miguel.valdes.s@mail.ucv.cl, claudio.cubillos@ucv.cl

Abstract. This work presents the results of designing a multiagent system for the tourism sector, whose objective is to give support to the tourist before and during his travel by means of mobile devices. The system focuses on the creation of itineraries, but besides allows the booking, publication and notification of changes in the services. The system was designed with the PASSI methodology and a prototype is being implemented over Jade.

Keywords: PASSI, Agents, Itinerary, Tourism, Mobile Devices.

1 Introduction

In order for the tourism industry would reach the success and development that allows it today to be one of the biggest world industries, much is due to the information technologies that from its beginnings has generated a wide range of information-technology systems that provide support for both, the tourist and the service providers. In that direction, the present project consists in designing a multi-agent system that allows the tourist to plan his travels by means of some mobile device. The system will have to support the tourist in the obtaining of an itinerary, provide a means to publish new services, allow the booking of these services and the notification of eventual changes on them.

This system is thought to be a large-scale system, that is, a multi-agent system that holds a large number of agents. First, we begun designing the system as from similar projects, next have identified some bottlenecks that were generated when agents' number is increased and when information processing consumes too many resources.

2 Related Work

In the last years have been developed diverse projects related with tourism, agents and mobile devices. Firstly, in [1] is presented the specification of a personal travel assistant that followed FIPA standards. Among the most relevant projects found in literature can be mentioned @lisTur-Móvil [4], Crumpet [5], GUIDE [6], and Palio [7]. However, in general terms these projects do not provide much information on its results. The only exception is the e-Travel [8] project of which even the source code is available. Other related works with tourism are numerous Web projects, but the use of

this technology forces the tourist to be connected every time that wants to manage the created itinerary. A last source has been the ontologies found in the Ontoselect [9] website, the ones that were used in part to define a more extensive ontology.

This work is, in a certain way, the continuation of the specification accomplished by FIPA's group, but increasing their functionalities and focused in the efficiency of the system in presence of a large number of agents, and the resources consumption for part of the mobile device of the tourist.

3 The PASSI Methodology

This project will make use of PASSI (Process for Agent Societies Specification and Implementation) as development methodology, which uses UML as modeling language. This methodology accomplishes the specification of a multi-agent system with a bottom-up approach, and gives support for the verification and validation of its models and specifications. PASSI has a toolkit [3] for IBM Rational Rose to support this methodology, besides allows the user to follow and adequately implement the PASSI phases thanks to the automatic compilation of diagrams. For a detailed description of PASSI five models and its respective steps, please refer to [2].

4 The Tourist Industry

Tourism is defined officially as the set of activities accomplished by people that travel to (and they remain in) a place for diverse motivations, such as pleasure, business or other purposes. Tourism is based on mobility, hence the supply and demand of services conforms a worldwide network, where production and distribution is based on cooperation. Furthermore, it is an industry based in confidence among parties (clients and providers) and the information available, as at the moment of deciding and acquiring tourist services, only the information about the product is available and not the product itself [10].

The following elements compose the tourism industry:

- The *tourist* is a person who travels and lodges out of his domicile, and whose stay requires of varied tourist services.
- The tourist *service* is an activity, accomplished in a determined place that allows satisfying the tourist needs during his stays. These can be divided into four sectors: Attractions, Accommodation, Alimentation and Transportation.
- The *provider* is the person or company that provides a tourist service.
- The *intermediary* is the role performed by the tour operators, among others, to bring the tourist closer to the services available.

4.1 The Planning of Travels

It is difficult to know how much money is going to be spending in a travel. For it, it is necessary to establish an initial budget and the number of persons to be traveling. Later, we have to establish the travel dates. Next, we have to make a decision about the cities or regions that are wished to be visited, taking into account the number of

days to stay at them. Later, the activities need to be selected to be held in each place. Knowing the days to be spent on each place, the accommodation needs to be defined and depending if this includes alimentation or not, we have to search for somewhere to eat. Subsequently, we have to select the mediums of transportation that will allow us to move between the cities and to get to the places of concern. Finally, we have to be provided with all the bookings that are necessary.

As it can be foretold, to design an itinerary is not a simple task. The travel packages offered by tour operators exist in order to support the tourist in speeding-up the building of an itinerary. In these packages, it is possible to integrate several tourist services under a single product; however the problem is that the majority of these packages present a limited flexibility.

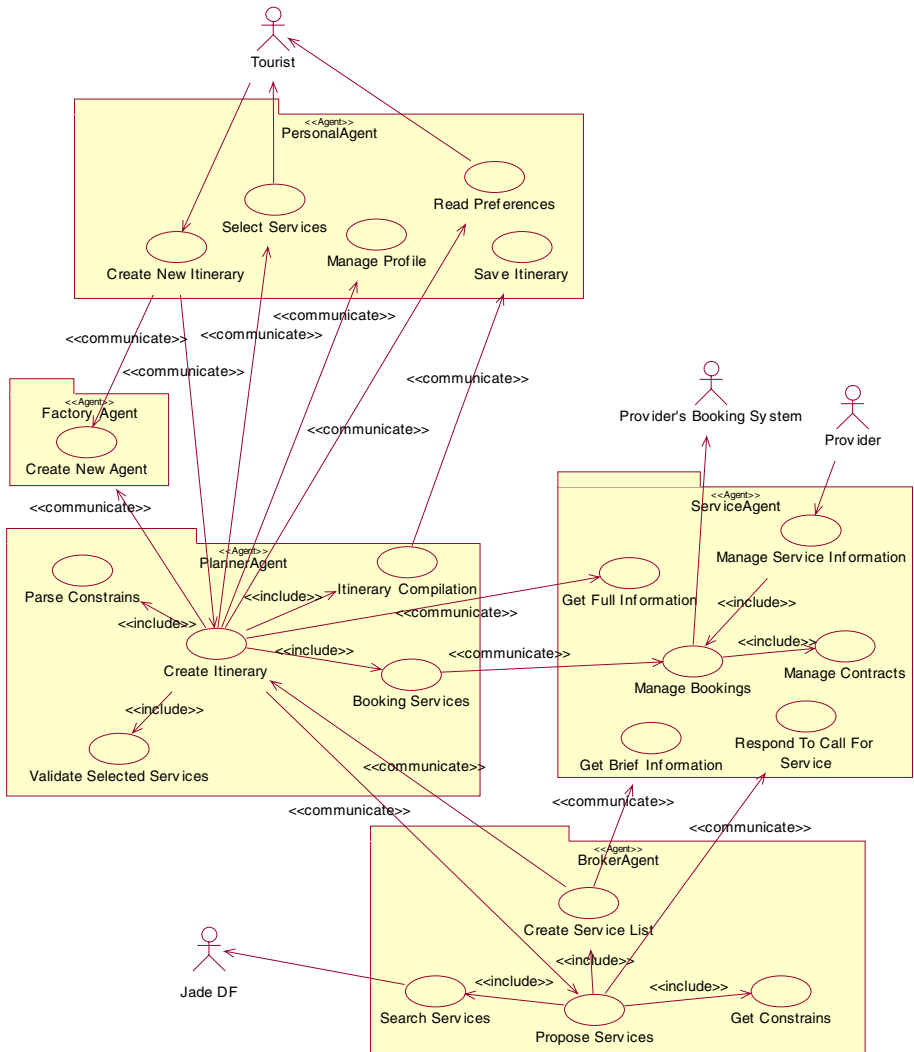


Fig. 1. Agent Identification Diagram

5 The Proposed Solution

Given the mobile nature of the tourist area, the proposed solution involves to design a system that adapts to diverse mobile devices, between these we can find PDAs, Mobile Phones, Smartphones, and traditional devices as a Laptops. In order to achieve this, will be designed a multi-agent system that permits the obtaining of itineraries from the tourist's preferences. Besides, to allow the personalization of the itinerary, the system will have to allow the booking of the selected services, notification of changes on them and remind the selected activities.

5.1 Multiagent System Design

The specification of this project was done using the PASSI methodology. Due to space reasons, will be showed only some figures.

From the System Requirements Model has been done the Domain Requirements Description diagram, and as of this diagram was grouped the system's functionalities to identifying the agents, resulting of this the Agent Identification diagram, of which is shown a simplified version in the Figure 2. The diagram shows the agents that will be a part of the system which are described in the following:

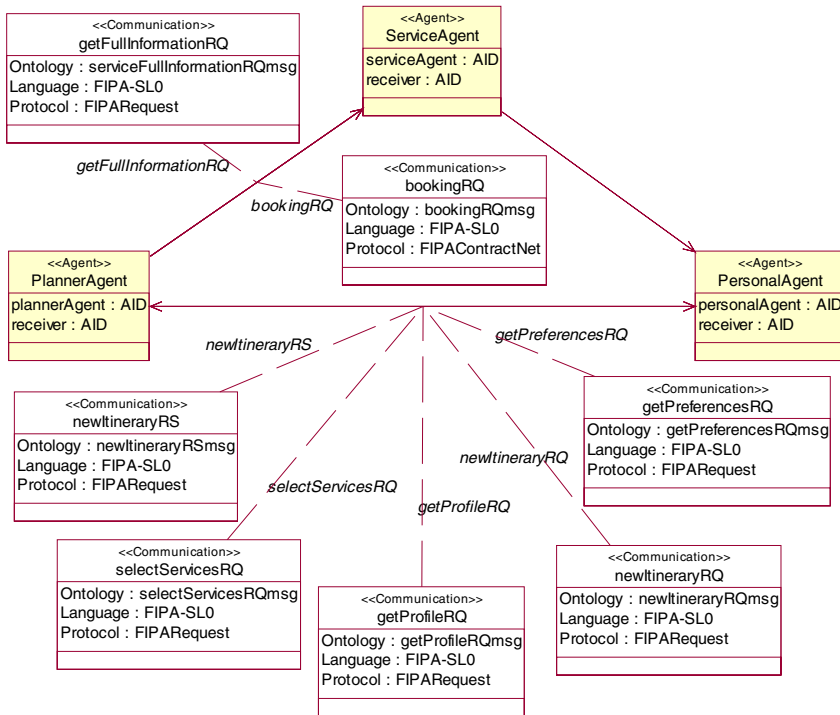


Fig. 2. Communication Ontology Description

Personal Agent: Is responsible to remember the tourist's profile, obey his instructions, manage the itinerary, search and notify changes in services. It is contingent upon the device in which it finds, such as a PDA or a Laptop. When this agent is created, could require the creation of a Planner Agent to accomplish some operations.

Planner Agent: Acts in the tourist's behalf. This agent should check the tourist's requirements, looking for the services through the Broker Agent, filtering them and booking the services selected by the tourist. This agent is independent from the tourist and when the agent finishes its execution, it does not save any information about the queries and is removed from the platform.

Broker Agent: Receives requirements from the Planner Agent to then proposing a list of services that can satisfy them. This agent will communicate with Jade's DF to get a list of registered services.

Service Agent: Is responsible of publishing and informing about the services that the provider offers. This agent manages information about the service that offers, its registration with the DF, and its booking if available, through a private Booking System. In the case of not having such a system, it will send the tourist the enough information to accomplish the booking by other means, such as phone or fax.

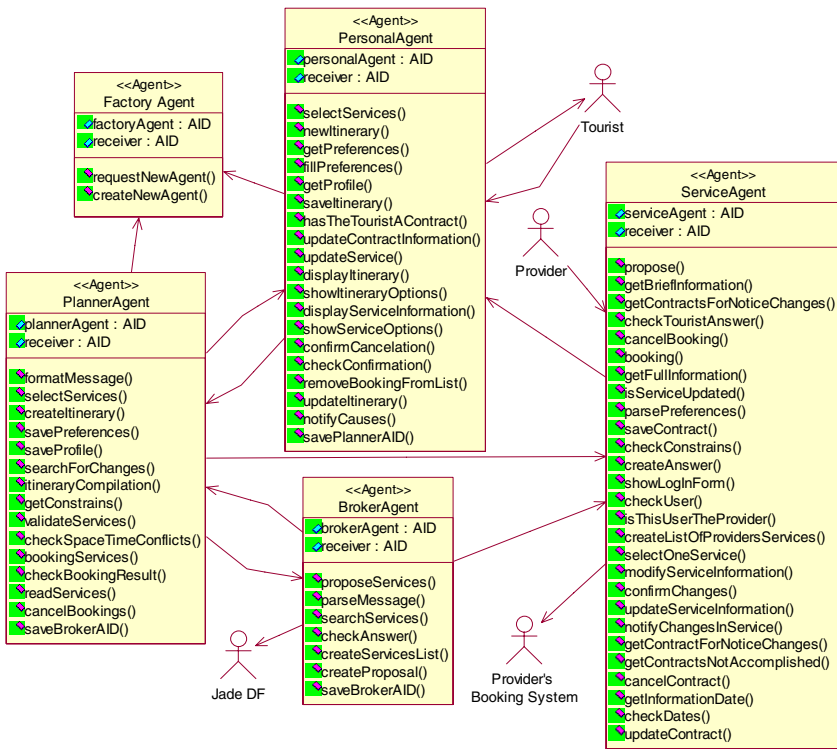


Fig. 3. Multiagent Structure Definition Diagram

Factory Agent: When a Personal Agent requires a Planner Agent, or a Planner Agent requires a Broker Agent, they request the Factory agent for the assignation of such agent. The Factory Agent will generate the agent when it is required and will send a message back with the created agent's AID.

From Agent Society Model has been done the Domain Ontology Description and Communication Ontology diagrams, of which can be appreciate a portion in Figure 2. In relation to Protocol Description, have to explain that with the protocols defined by FIPA satisfies the system's needs. Between these protocols, have to highlight the utilization of Contract Net for side of the Planner Agent, for the contract/booking of services. And the use of FIPA SLO as the language for communications.

The Multiagent Structure Definition diagram is presented in Figure 3, where can be appreciated the diverse actors and their relations with all the system's agents, relations that indicate the flow of existing information in the communications between agents. This diagram puts special emphasis in showing the agents' knowledge as attributes and showing their tasks as procedures.

6 Prototype Implementation

To test the accomplished agent architecture we have been working in a prototype using JADE/LEAP. In Figure 4 can be appreciated the Main Container's GUI that in addition to the presence of the three platform agents (*ams*, *df*, and *rma*) holds the *fa* agent that corresponds to the Factory Agent (loaded along with the platform). Can be appreciated also a container named *BE* which corresponds to the BackEnd that is necessary to be able to load the agents programmed with CLDC. This container holds the *pa* agent that corresponds to the agent loaded by the emulated mobile phone. This agent requested a Planner Agent named *pa_0*, loaded in the Main Container.

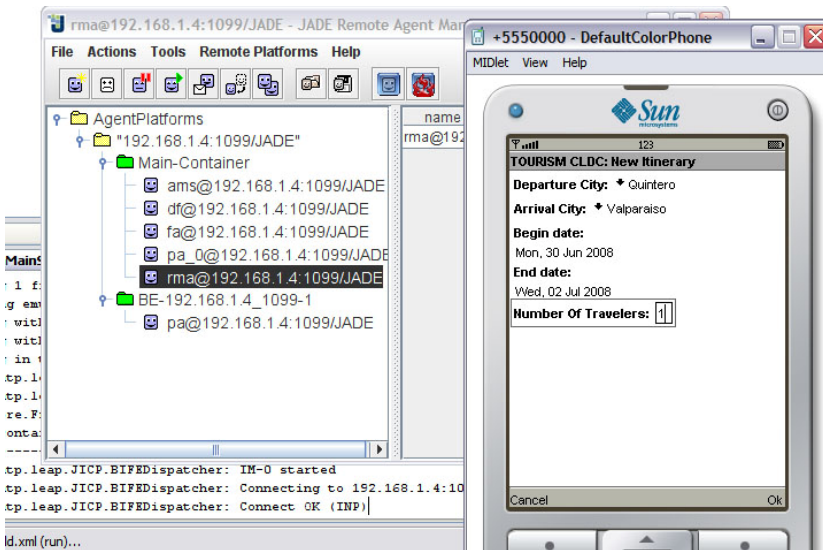


Fig. 4. Working prototype to test the design

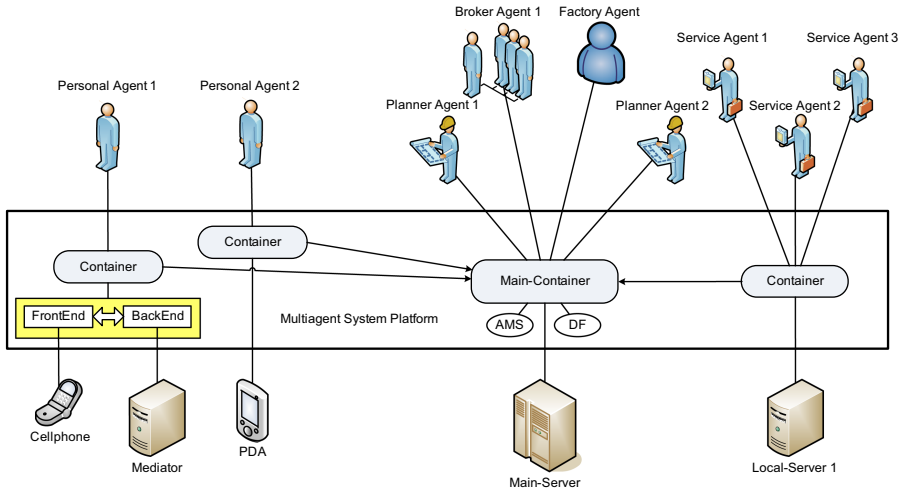


Fig. 5. Multiagent System Platform

At the right-hand can be seen the Personal Agent accomplishing the task of filling the form with the travel preferences. In this first screen should be selected the city of origin and destination, the beginning and ending travel-dates, and the number of travelers.

6.1 Agent's Deployment in the Platform

When an agent is launched in the platform, must do it on a Container, which provides an environment of execution to the agent. The system will consist of (see Figure 5) a set of containers so that the workload and connectivity to mobile devices would be low, and allows the distribution of services through various local containers.

7 Conclusions and Future Work

The present work has depicted an agent architecture devoted to the tourist domain. In particular the system allows tourists to arrange itineraries through the use of mobile devices.

As future work will be continued the work on the prototype to test the validity of the accomplished design. Will be important to study the quantity of information that will be able to keep in the device the Personal Agent. The system's performance when exist a large number of agents making an offer and requesting services. The overload that the system will have while creating agents when are required and evaluating the possibility of hold an minimum number of agents waiting.

Acknowledgements. The present work has been partially funded by Grant PUCV 037.215/2008 under the "Collaborative Systems" Nucleus Project.

References

1. FIPA Architecture Board.: FIPA Personal Travel Assistance Specification FIPA (2001)
2. Cossentino, M., Potts, C.: PASSI: a Process for Specifying and Implementing Multi-Agent Systems Using UML (2001)
3. PASSI Toolkit (PTK), <http://sourceforge.net/projects/ptk>
4. Raventós, G., et al.: Turismo y Patrimonio Cultural Latinoamericano: Una Aplicación de Comercio Electrónico Agentes Inteligentes. In: XI Convención Informática 2005, La Habana, Cuba (2005)
5. Poslad, S., et al.: Crumppet: Creation of user-friendly mobile services personalised for tourism. In: Proceedings of 3G 2001 (2001)
6. Davies, N., et al.: Developing a Context Sensitive Tourist Guide, Technical Report Computing Department, Lancaster University (March 1998)
7. Zarikas, V., et al.: An architecture for a self-adapting information system for tourists. Proc. 2001 (2001)
8. Kalczyński, P., et al.: Personalized Traveler Information System (2001)
9. Ontoselect, <http://olp.dfki.de/ontoselect/>
10. Staab, S., et al.: Intelligent Systems for Tourism. IEEE Educational Activities Department (2002)

Author Index

- Abbas, Safia 251
Alhashmi, Saadat M. 14, 259
Amouroux, Edouard 26
Arai, Sachiyo 34
Arbab, Farhad 42
Aştefănoaei, Lăcrămioara 42, 54
Auger, Pierre 295
- Beydoun, Ghassan 98
Boella, Guido 66, 78, 86
Bogg, Paul 98
Botti, Vicente 197
Boucher, Alain 127
Bourgne, Gauvain 109
Broersen, Jan 86
Bui, The Duy 222
- Cabrera-Paniagua, Daniel 121
Chu, Thanh-Quang 127
Cubillos, Claudio 121, 397
- Dastani, Mehdi 42, 54, 139
de Boer, Frank S. 42, 54
Desvaux, Stéphanie 26
Dinh, Huy Q. 153
Do, Nguyen Luong 222
Do Duc, Dong 153
Drogoul, Alexis 1, 26, 127, 295
Duc, Nguyen Tuan 307
- Ehlers, Elizabeth Marie 340
- Fujita, Katsuhide 161
Fukuta, Naoki 173
Furuhata, Masabumi 185
- Garcia, Emilia 197
Giret, Adriana 197
Governatori, Guido 315, 328
Gutiérrez-García, J. Octavio 206
- Hassan, Mohd Fadzil 214
Hattori, Hiromitsu 370
Hirata, Hironori 34
Ho, Dac Phuong 222
- Ho, Tuong Vinh 287
Hoang, Tuan Nha 389
Hoang Xuan, Huan 153
- Ishida, Toru 370
Ishigaki, Yoshihisa 34
Ito, Takayuki 161, 173, 231
Iwakami, Masashi 231
- Jamroga, Wojciech 239
- Khayyambashi, Mohammad-Reza 6
Kim, Minkoo 267
Klein, Mark 161
Koning, Jean-Luc 206
Kuribara, Shusuke 251
- Lam, YiHua 259
Lee, Keonsoo 267
Lopez-Carmona, Miguel A. 275
Low, Graham 98
- Mahmood, Ahmad Kamil 357
Marilleau, Nicolas 287
Marsa-Maestre, Ivan 275
Maudet, Nicolas 109
McCartney, Robert 323
Meyer, John-Jules 42, 54
Mol, Christian P. 139
- Nakajima, Kengo 370
Nguyen, Hong-Phuong 127
Nguyen, Ngoc Doanh 295
Nguyen, Ngoc Thanh 2
Nguyen, Trong Khanh 287
Noorae Abadeh, Maryam 6
Novani, Santi 348
- Odagaki, Marika 370
- Padgham, Lin 4
Padmanabhan, Vineet 315
Park, Gi-Duck 323
Perrussel, Laurent 185
Pham, Duy Hoang 328
Pike, Janine Claire 340
Putro, Utomo Sarjono 348

- Rahman, Arief 357
Ramos-Corchado, Félix F. 206
Robertson, Dave 214

Sawada, Shoichi 370
Sawamura, Hajime 251
Schneider, Etienne 357
Seghrouchni, Amal El Fallah 109
Siallagan, Manahan 348
Soldano, Henry 109
Steunebrink, Bas R. 139

Takeuchi, Ikuo 307
Tan, Vu Van 381
Thakur, Subhasis 315, 328
Tinnermeier, Nick 42
Tran, Dinh Que 389

Utomo, Dhanan Sarwo 348

Valdés, Miguel 397
van der Torre, Leendert 66, 78, 86
Velasco, Juan R. 275
Villata, Serena 66, 78
Vo, Duc-An 127

Yang, Jung-Jin 323
Yi, Myeong-Jae 381
Yoo, Dae-Seung 381

Zaminifar, Kamran 6
Zhang, Dongmo 185
Zucker, Jean-Daniel 127